IBM

Systems Journal

–1988 CUMULATIVE–

INDEX

–AUTHORS–SUBJECTS–ABSTRACTS–

Editorial Staff

Editor
G. M. Gershon

Associate Editors
A. G. Davis
M. J. Haims
G. C. Stierhoff

Editorial Associate
C. R. Seddon

Editorial Coordinator
L. P. Hudd

Editorial Secretary
K. M. Burg

Publications Staff

Publications Manager
C. E. Tangney

Art Director
J. F. Musgrave

Production Assistant
L. A. Fasone

Editorial Assistant
A. R. Thornton

Advisory Board

J. A. Armstrong
R. P. Case
G. H. Conrades
C. J. Conti
N. M. Donofrio
H. G. Figueroa
R. E. Gomory
E. M. Hancock
B. Reisman
S. B. Schwartz
P. A. Toole
E. F. Wheeler

Director of Technical Journals

W. J. Turner

Using the IBM Systems Journal Index

This index consists of three parts. First is the Author Index which lists all authors alphabetically from the inception of the *IBM Systems Journal* in 1962 through 1988 and includes complete titles of all of the papers. Second is the Subject Index which categorizes each paper according to one or more of 26 topical areas. This part includes papers published from 1972 through 1988. Last is a list of abstracts for papers published from 1981 through 1988 by volume number and issue.

In the Author Index, the first item of an entry is the name of an author with last name given first. If a paper has multiple authors, the entry containing the title of the paper includes all of the authors. The additional authors are also listed in individual entries.

A sample from the Author Index is shown below in which the first entry lists the names of the authors first. Next to be seen is the title in italics, followed by the volume number (in boldface type), the issue number, the number of the page on which the paper begins, and the year of publication. Beginning with papers published in 1975, an order number for the reprint of the paper is given. It appears in italics as the last item. The second entry depicts the way in which other authors for multiple-authored papers are listed.

> Ambrosetti, R., T. A. Ciriani, and R. Pennachi, *An application analyzer* **23**, 4, 336 (1984); *G321-5229.*
> Ciriani, T. A., see Ambrosetti, R., **23**, 4, 336 (1984).

Entries in the Subject Index list the title of a paper first, followed by the last name of the author. For papers that have multiple authors, the name given is that of the author whose name appears first on the paper. The next item listed is the volume number in boldface type, followed by the issue number, the page number of the first page of the paper, and the year of publication.

In the Abstracts section, papers are listed chronologically by volume, issue, and year and in the order in which they appear in the table of contents in each issue. The item shown in italics after the year is the order number for the issue. Individual entries list the title first in boldface, followed by the authors and the page on which the paper begins in italics.

The *IBM Systems Journal* is published quarterly by International Business Machines Corporation, Armonk, NY 10504. Officers: John F. Akers, Chairman of the Board; David A. Finley, Treasurer; William W. K. Rich, Secretary.

Subscriptions to the *IBM Systems Journal* are $20.00 per calendar year. To order subscriptions or report change of address, write to IBM, P.O. Box 2076, Mahopac, NY 10541. Additional information about ordering reprints and complete issues is given on the inside back cover. Manuscript submissions and all other correspondence should be addressed to the Editor, *IBM Systems Journal*, Armonk, NY 10504.

Subscriptions may be entered or cancelled by IBM employees in the U.S.A. via VM/CMS by using the command JOURNALS, or by mailing the Journals Subscription Card (ZMO8-2203). Change of address for IBM employees is handled automatically by the IBM Corporate Employee Resource Information System and need not be reported to the *Journal*. IBM employees outside the U.S.A. may subscribe through their Country Literature Coordinators or local IBM libraries.

IBM Systems Journal Author Index 1962–1988

Acker, R. D., and P. H. Seaman, *Modeling distributed processing across multiple CICS/VS sites* **21**, 4, 471 (1982); *G321-5177.*

Agassi, S., *Performance considerations for a distributed data processing system designed for high availability* **24**, 3/4, 200 (1985); *G321-5249.*

Ahmadi, M., J. H. Chou, and G. Gafka, *NetView/PC* **27**, 1, 32 (1988); *G321-5309.*

Ahuja, D. V., and S. A. Coons, *Interactive Graphics in Data Processing Geometry for construction and display* **7**, 3/4, 188 (1968).

Ahuja, D. V., *Interactive Graphics in Data Processing An algorithm for generating spline-like curves* **7**, 3/4, 206 (1968).

Ahuja, V., *Routing and flow control in Systems Network Architecture* **18**, 2, 298 (1979); *G321-5097.*

Ahuja, V., *Common Communications Support in Systems Application Architecture* **27**, 3, 264 (1988); *G321-5324.*

Aiso, M., *Forecasting techniques* **12**, 2, 187 (1973).

Albrecht, H. R., and K. D. Ryder, *The Virtual Telecommunications Access Method: A Systems Network Architecture perspective* **15**, 1, 53 (1976); *G321-5027.*

Albrecht, H. R., and L. C. Thomason, *I/O facilities of the Distributed Processing Programming Executive (DPPX)* **18**, 4, 526 (1979); *G321-5108.*

Alfonseca, M., M. L. Tavera, and R. Casajuana, *An APL interpreter and system for a small computer* **16**, 1, 18 (1977); *G321-5043.*

Alfonseca, M., see Tavera, M. L., **24**, 1, 61 (1985).

Allen, A. O., *Elements of probability for system design* **13**, 4, 325 (1974).

Allen, A. O., *Elements of queuing theory for system design* **14**, 2, 161 (1975); *G321-5009.*

Allen, C. D., *The application of formal logic to programs and programming* **10**, 1, 2 (1971).

Almond, D. B., *Numerical control for machining complex surfaces* **11**, 2, 150 (1972).

Altman, E. B., see Senko, M. E., **12**, 1, 30 (1973).

Altman, E. B., see Senko, M. E., **12**, 1, 45 (1973).

Altman, E. B., see Senko, M. E., **12**, 1, 64 (1973).

Ambrosetti, R., T. A. Ciriani, and R. Pennacchi, *An application analyzer* **23**, 4, 336 (1984); *G321-5229.*

Amdahl, G. M., *The structure of SYSTEM/360 Part III: Processing unit design considerations* **3**, 2, 144 (1964).

Amy, R. M., see Sundstrom, R. J., **26**, 1, 13 (1987).

Anastassiou, D., M. K. Brown, H. C. Jones, J. L. Mitchell, W. B. Pennebaker, and K. S. Pennington, *Series/1-based videoconferencing system* **22**, 1/2, 97 (1983); *G321-5185.*

Anderson, H. A., Jr., M. Reiser, and G. L. Galati, *Tuning a virtual storage system* **14**, 3, 246 (1975); *G321-5013.*

Anderson, J. W., see Singh, Y., **25**, 1, 20 (1986).

Angeleri, P., D. F. Lozupone, F. Piccolo, and J. Clinckemaillie, *PAM-CRASH on the IBM 3090/VF: An integrated environment for crash analysis* **27**, 4, 541 (1988); *G321-5342.*

Anthony, A. L., and H. K. Watson, *Techniques for developing analytic models* **11**, 4, 316 (1972).

Antonelli, D. C., *The role of the operator in the Supermarket and Retail Store Systems* **14**, 1, 35 (1975); *G321-5003.*

Anzick, W. J., see Mendelsohn, N., **22**, 1/2, 63 (1983).

Appel, A., T. P. Dankowski, and R. L. Dougherty, *Interactive Graphics in Data Processing Aspects of display technology* **7**, 3/4, 176 (1968).

Appel, A., *Interactive Graphics in Data Processing Modeling in three dimensions* **7**, 3/4, 310 (1968).

ApSimon, H. G., *Algorithm for a gear-train problem* **3**, 1, 95 (1964).

Archibald, J. L., B. M. Leavenworth, and L. R. Power, *Abstract design and program translator: New tools for software design* **22**, 3, 170 (1983); *G321-5189.*

Arnold, J. S., D. P. Casey, and R. H. McKinstry, *Design of tightly-coupled multiprocessing programming* **13**, 1, 60 (1974).

Aron, J. D., *Real-time systems in perspective* **6**, 1, 49 (1967).

Aschenbrenner, J. R., *Open Systems Interconnection* **25**, 3/4, 369 (1986); *G321-5281.*

Ascoly, J., see Kuki, H., **10**, 1, 39 (1971).

Asher, J. R., see Skinner, C. E., **8**, 4, 319 (1969).

Ashford, T. J., see Burns, N. A., **25**, 2, 236 (1986).

Astrahan, M. M., see Senko, M. E., **12**, 1, 30 (1973).

Astrahan, M. M., see Senko, M. E., **12**, 1, 45 (1973).

Astrahan, M. M., see Senko, M. E., **12**, 1, 64 (1973).

Astrahan, M. M., see Blasgen, M. W., **20**, 1, 41 (1981).

Attanasio, C. R., P. W. Markstein, and R. J. Phillips, *Penetrating an operating system: a study of VM/370 integrity* **15**, 1, 102 (1976); *G321-5029.*

Attanasio, C. R., *Virtual Control Storage–security measures in VM/370* **18**, 1, 93 (1979); *G321-5088.*

Auslander, M. A., and J. F. Jaffe, *Functional structure of IBM virtual storage operating systems Part I: Influences of dynamic address translation on operating system technology* **12**, 4, 368 (1973).

Bailey, R. M., and R. C. Soucy, *Performance and availability measurement of the IBM Information Network* **22**, 4, 404 (1983); *G321-5202.*

Baily, J. H., J. A. Howard, and T. J. Szczygielski, *The job entry subsystem of OS/VS1* **13**, 3, 253 (1974).

Baker, C. T., *Fabrication and assembly operations Part I: The outlines of a control system* **4**, 2, 87 (1965).

Baker, C. T., *Logical distribution of applications and data* **19**, 2, 171 (1980); *G321-5121.*

Baker, F. T., *Chief programmer team management of production programming* **11**, 1, 56 (1972).

Baldwin, F. R., W. B. Gibson, and C. B. Poland, *A multiprocessing approach to a large computer system* **1**, 1, 64 (1962).

Ball, F. W., see Sakamoto, J. G., **21**, 1, 54 (1982).

Banham, J. A., and P. McClelland, *Design features of a real-time check-clearing system* **11**, 4, 329 (1972).

Baratz, A. E., see Sultan, R. A., **26**, 4, 429 (1987).

Bard, Y., *Performance criteria and measurement for a time-sharing system* **10**, 3, 193 (1971).

Bard, Y., *Experimental evaluation of system performance* **12**, 3, 302 (1973).

Bard, Y., *Performance analysis of virtual memory time-sharing systems* **14**, 4, 366 (1975); *G321-5022.*

Barzilai, T. P., see Sultan, R. A., **26**, 4, 429 (1987).

Baskin, H. B., and S. P. Morse, *Interactive Graphics in Data Processing A multilevel modeling structure for interactive graphic design* **7**, 3/4, 218 (1968).

Belady, L. A., *A study of replacement algorithms for a virtual-storage computer* **5**, 2, 78 (1966).

Belady, L. A., M. W. Blasgen, C. J. Evangelisti, and R. D. Tennison, *A computer graphics system for block diagram problems* **10**, 2, 143 (1971).

Belady, L. A., and M. M. Lehman, *A model of large program development* **15**, 3, 225 (1976); *G321-5035.*

Belady, L. A., C. J. Evangelisti, and L. R. Power, *GREENPRINT: A graphic representation of structured programs* **19**, 4, 542 (1980); *G321-5137.*

Bender, G., D. N. Freeman, and J. D. Smith, *Function and design of DOS/360 and TOS/360* **6**, 1, 2 (1967).

Benjamin, J. H., M. L. Hess, R. A. Weingarten, and W. R. Wheeler, *Interconnecting SNA networks* **22**, 4, 344 (1983); *G321-5199.*

Bennett, J. L., see Bullen, C. V., **21**, 3, 351 (1982).

Bennett, J. L., *Tools for building advanced user interfaces* **25**, 3/4, 354 (1986); *G321-5280.*

Beregi, W. E., *Architecture prototyping in the software engineering environment* **23**, 1, 4 (1984); *G321-5206.*

Beregi, W. E., see Hoffnagle, G. F., **24**, 2, 102 (1985).

Beretvas, T., *Performance tuning in OS/VS2 MVS* **17**, 3, 290 (1978); *G321-5077.*

Beretvas, T., see Tetzlaff, W. H., **26**, 2, 215 (1987).

Berinato, C. L., see deBry, R. K., **27**, 2, 234 (1988).

Berk, M. A., C. W. Dunbar, and G. C. Hobson, *Design and performance considerations for the Retail Store System* **14**, 1, 64 (1975); *G321-5005.*

Berry, R. E., *Common User Access–A consistent and usable human-computer interface for the SAA environments* **27**, 3, 281 (1988); *G321-5325.*

Bertrand, O. P., see Chamberlin, D. D , **21**, 3, 250 (1982).

Bethke, F. J., W. M. Dean, P. H. Kaiser, E. Ort, and F. H. Pessin, *Improving the usability of programming publications* **20**, 3, 306 (1981); *G321-5151.*

Bharath-Kumar, K., see Grover, G. A., **22**, 4, 451 (1983).

Bias, R. G., see Waters, F. C. H., **26**, 4, 383 (1987).

Birch, J. P., *Functional structure of IBM virtual storage operating systems Part III: Architecture and design of DOS/VS* **12**, 4, 401 (1973).

Bird, D. L., and C. U. Munoz, *Automatic generation of random self-checking test cases* **22**, 3, 229 (1983); *G321-5193.*

Bird, R. A., and C. A. Hofmann, *Systems management* **19**, 1, 140 (1980); *G321-5120.*

Bitonti, F., D. W. Cooper, D. N. Frayne, and H. H. Hansen, *A computer-aided linkage analysis system* **4**, 3, 200 (1965).

Bjork, L. A., Jr., *Generalized audit trail requirements and concepts for data base applications* **14**, 3, 229 (1975); *G321-5012.*

Blaauw, G. A., and F. P. Brooks, Jr., *The structure of SYSTEM/360 Part I: Outline of the logical structure* **3**, 2, 119 (1964).

Blaauw, G. A., *The structure of SYSTEM/360 Part V: Multisystem organization* **3**, 2, 181 (1964).

Black, B. C., and D. C. Gazis, *Real-time traffic flow optimization* **10**, 3, 217 (1971).

Blake, K., and G. Gordon, *Systems simulation with digital computers* **3**, 1, 14 (1964).

Blakeney, G. R., L. F. Cudney, and C. R. Eickhorn, *An application-oriented multiprocessing system Part II: Design characteristics of the 9020 system* **6**, 2, 80 (1967).

Blasgen, M. W., see Belady, L. A., **10**, 2, 143 (1971).

Blasgen, M. W., and K. P. Eswaran, *Storage and access in relational data bases* **16**, 4, 363 (1977); *G321-5058.*

Blasgen, M. W., M. M. Astrahan, D. D. Chamberlin, J. N. Gray, W. F. King, B. G. Lindsay, R. A. Lorie, J. W. Mehl, T. G. Price, G. R. Putzolu, M. Schkolnick, P. G. Selinger, D. R. Slutz, H. R. Strong, I. L. Traiger, B. W. Wade, and R. A. Yost, *System R: An architectural overview* **20**, 1, 41 (1981); *G321-5140.*

Bleher, J. H., P. G. Caspers, H. H. Henn, and K. Maerker, *A graphic interactive application monitor* **19**, 3, 382 (1980); *G321-5132.*

Blum, A. M., *A general purpose digital simulator and examples of its application Part III: Digital simulation of urban traffic* **3**, 1, 41 (1964).

Boies, S. J., *User behavior on an interactive computer system* **13**, 1, 2 (1974).

Boies, S. J., see Gould, J. D., **23**, 1, 65 (1984).

Bomberault, A. M., see White, W. W., **8**, 2, 147 (1969).

Bonner, A. J., *Using system monitor output to improve performance* **8**, 4, 290 (1969).

Boos, W., *HONE: The IBM marketing support system* **24**, 3/4, 189 (1985); *G321-5248.*

Botterill, J. H., *The design rationale of the System/38 user interface* **21**, 4, 384 (1982); *G321-5174.*

Boyd, D. F., and H. S. Krasnow, *Economic evaluation of management information systems* **2**, 1, 2 (1963).

Boyd, D. F., H. S. Krasnow, and A. C. R. Petit, *A general purpose digital simulator and examples of its application Part IV: Simulation of an integrated steel mill* **3**, 1, 51 (1964).

Bozman, G., W. Buco, T. P. Daly, and W. H. Tetzlaff, *Analysis of free-storage algorithms–revisited* **23**, 1, 44 (1984); *G321-5209.*

Branin, F. H., Jr., L. V. Hall, J. Suez, R. M. Carlitz, and T. C. Chen, *An interpretive program for matrix arithmetic* **4**, 1, 2 (1965).

Branscomb, L. M., *Computing and communications–A perspective of the evolving environment* **18**, 2, 189 (1979); *G321-5092.*

Branscomb, L. M., and J. C. Thomas, *Ease of use: A system design challenge* **23**, 3, 224 (1984); *G321-5220.*

Brennan, R. D., and M. Y. Silberberg, *Two continuous system modeling programs* **6**, 4, 242 (1967).

Bresenham, J. E., *Algorithm for computer control of a digital plotter* **4**, 1, 25 (1965).

Brewer, A. C., *Interactive scheduling system* **10**, 1, 62 (1971).

Bricault, J. P., and I. Delgalvis, *On teleprocessing system design Part III: An analysis of a request-queued buffer pool* **5**, 3, 148 (1966).

Brockish, R. F., see Schick, T., **21**, 2, 220 (1982).

Bronner, L., *Overview of the capacity planning process for production data processing* **19**, 1, 4 (1980); *G321-5113.*

Brooks, F. P., Jr., see Blaauw, G. A., **3**, 2, 119 (1964).

Brooks, R. C., *An approach to high availability in high-transaction-rate systems* **24**, 3/4, 279 (1985); *G321-5254.*

Brown, D. T., R. L. Eibsen, and C. A. Thorn, *Channel and direct access device architecture* **11**, 3, 186 (1972).

Brown, J. F., and D. W. Low, *Interactive simulation for banking* **12**, 2, 172 (1973).

Brown, M. K., see Anastassiou, D., **22**, 1/2, 97 (1983).

Brown, S. H., *Interactive Graphics in Data Processing Conversational job control* **7**, 3/4, 271 (1968).

Bryant, P., F. W. Giesin, Jr., and R. M. Hayes, *Experiments in line quality monitoring* **15**, 2, 124 (1976); *G321-5030.*

Buchholz, W., *File organization and addressing* **2**, 2, 86 (1963).

Buchholz, W., *A synthetic job for measuring system performance* **8**, 4, 309 (1969).

Buchholz, W., *The IBM System/370 vector architecture* **25**, 1, 51 (1986); *G321-5261.*

Buchwald, L. A., R. W. Davison, and W. P. Stevens, *Integrating applications with SAA* **27**, 3, 315 (1988); *G321-5328.*

Buckelew, B. R., *The System Planning Grid: A model for building integrated information systems* **24**, 3/4, 294 (1985); *G321-5255.*

Buco, W., see Bozman, G., **23**, 1, 44 (1984).

Buco, W. M., see Tetzlaff, W. H., **23**, 4, 375 (1984).

Buco, W. M., see Tetzlaff, W. H., **26**, 2, 215 (1987).

Bullen, C. V., J. L. Bennett, and E. D. Carlson, *A case study of office workstation use* **21**, 3, 351 (1982); *G321-5173.*

Burchi, R. S., *Interactive graphics today* **19**, 3, 292 (1980); *G321-5126.*

Burns, N. A., T. J. Ashford, C. T. Iwaskiw, R. P. Starbird, and R. L. Flagg, *The Portable Inference Engine: Fitting significant expertise into small systems* **25**, 2, 236 (1986); *G321-5273.*

Butler, C. W., see Hodil, E. D., **25**, 2, 147 (1986).

Bux, W., *Performance issues in local-area networks* **23**, 4, 351 (1984); *G321-5230.*

Byrd, R. J., see Heidorn, G. E., **21**, 3, 305 (1982).

Calica, A. B., *Fabrication and assembly operations Part II: Long-range planning techniques* **4**, 2, 94 (1965).

Calica, A. B., *Fabrication and assembly operations Part V: Production order sequencing* **4**, 3, 225 (1965).

Callaway, P. H., J. P. Considine, and C. H. Thompson, *Uses of virtual storage systems in a scientific environment* **11**, 3, 200 (1972).

Callaway, P. H., *Performance measurement tools for VM/370* **14**, 2, 134 (1975); *G321-5008.*

Campenni, R. D., see Leach, J. R., **19**, 1, 120 (1980).

Carlitz, R. M., see Branin, F. H., Jr., **4**, 1, 2 (1965).

Carlson, E. D., see Weller, D. L., **19**, 3, 314 (1980).

Carlson, E. D., see Bullen, C. V., **21**, 3, 351 (1982).

Carpenter, M. B., and H. K. Hallman, *Quality emphasis at IBM's Software Engineering Institute* **24**, 2, 121 (1985); *G321-5243.*

Carroll, J. M., see Thomas, J. C., **20**, 2, 237 (1981).

Casajuana, R., see Alfonseca, M., **16**, 1, 18 (1977).

Casey, C. A., see Griffee, A. W., **27**, 2, 206 (1988).

Casey, D. P., see Arnold, J. S., **13**, 1, 60 (1974).

Caspers, P. G., see Bleher, J. H., **19**, 3, 382 (1980).

Cawsey, G. P., see Wade, R. D., **8**, 1, 28 (1969).

Cenfetelli, A. R., *Data management concepts for DOS/360 and TOS/360* **6**, 1, 22 (1967).

Chamberlin, D. D., see Blasgen, M. W., **20**, 1, 41 (1981).

Chamberlin, D. D., O. P. Bertrand, M. J. Goodfellow, J. C. King, D. R. Slutz, S. J. P. Todd, and B. W. Wade, *JANUS: An interactive document formatter based on declarative tags* **21**, 3, 250 (1982); *G321-5169.*

Chambers, F. W., see Cole, H., **2**, 3/4, 240 (1963).

Chang, P. Y., and W. W. Myre, *OS/2 EE Database Manager overview and technical highlights* **27**, 2, 105 (1988); *G321-5313.*

Chang, W., and D. J. Wong, *Computer channel interference analysis* **4**, 2, 162 (1965).

Chang, W., *A queuing model for a simple case of time sharing* **5**, 2, 115 (1966).

Chang, W., *Single-server queuing processes in computing systems* **9**, 1, 36 (1970).

Chappuis, R. J., see Green, P. E., Jr., **26**, 4, 414 (1987).

Chen, F. C., and R. L. Dougherty, *Interactive Graphics in Data Processing A system for implementing interactive applications* **7**, 3/4, 257 (1968).

Chen, R., H. Crowder, and E. L. Johnson, *Solving the installation scheduling problem using mixed integer linear programming* **17**, 1, 82 (1978); *G321-5065.*

Chen, T. C., see Branin, F. H., Jr., **4**, 1, 2 (1965).

Cheng, J. M., C. R. Loosley, A. Shibamiya, and P. S. Worthington, *IBM Database 2 performance: Design, implementation, and tuning* **23**, 2, 189 (1984); *G321-5218.*

Cheng, P. S., *Trace-driven system modeling* **8**, 4, 280 (1969).

Chess, D. M., and G. Waldbaum, *The VM/370 Resource Limiter* **20**, 4, 424 (1981); *G321-5157.*

Chess, D. M., *A tight coupling of workstations* **23**, 3, 255 (1984); *G321-5223.*

Chess, D. M., and M. F. Cowlishaw, *A large-scale computer conferencing system* **26**, 1, 138 (1987); *G321-5291.*

Chien, R. T., see Tang, D. T., **8**, 1, 48 (1969).

Chin, S. T., see Fleischmann, A., **26**, 3, 255 (1987).

Chiu, W. W., and W. M. Chow, *A performance model of MVS* **17**, 4, 444 (1978); *G321-5083.*

Chodorow, M. S., see Heidorn, G. E., **21**, 3, 305 (1982).

Chou, J. H., see Ahmadi, M., **27**, 1, 32 (1988).

Chow, W. M., see Chiu, W. W., **17**, 4, 444 (1978).

Choy, D. M., see Lum, V. Y., **21**, 3, 327 (1982).

Christensen, K., G. P. Fitsos, and C. P. Smith, *A perspective on software science* **20**, 4, 372 (1981); *G321-5154.*

Church, F. L., *Requirements generation, explosions, and bills of material* **2**, 3/4, 268 (1963).

Ciarfella, W. A., see Radice, R. A., **24**, 2, 79 (1985).

Ciarfella, W. A., see Mays, R. G., **24**, 2, 134 (1985).

Ciriani, T. A., see Ambrosetti, R., **23**, 4, 336 (1984).

Clark, I. A., *Software simulation as a tool for usable product design* **20**, 3, 272 (1981); *G321-5149.*

Clark, R. S., and T. L. Wilson, *Vector system performance of the IBM 3090* **25**, 1, 63 (1986); *G321-5262.*

Clark, W. A., *The functional structure of OS/360 Part III: Data management* **5**, 1, 30 (1966).

Clarkson, T. B., III, *Standardized graphics on the IBM Personal Computer* **24**, 1, 3 (1985); *G321-5233.*

Clementi, E., D. Logan, and J. Saarinen, *ICAP 3090: Parallel processing for large-scale scientific and engineering problems* **27**, 4, 475 (1988); *G321-5339.*

Clinckemaillie, J., see Angeleri, P., **27**, 4, 541 (1988).

Cole, H., Y. Okaya, and F. W. Chambers, *A computer-operated laboratory data-taking system* **2**, 3/4, 240 (1963).

Cole, H., *System/7 in a hierarchical laboratory automation system* **13**, 4, 307 (1974).

Cole, H., and A. A. Guido, *The IBM 5100 and the Research Device Coupler–A personal laboratory automation system* **16**, 1, 41 (1977); *G321-5044.*

Cole, L. J., see Sundstrom, R. J., **26**, 1, 13 (1987).

Comba, P. G., *Interactive Graphics in Data Processing A language for three-dimensional geometry* **7**, 3/4, 292 (1968).

Considine, J. P., see Callaway, P. H., **11**, 3, 200 (1972).

Considine, J. P., and J. J. Myers, *MARC: MVS archival storage and recovery program* **16**, 4, 378 (1977); *G321-5059.*

Constantine, L. L., see Stevens, W. P., **13**, 2, 115 (1974).

Conti, C. J., D. H. Gibson, and S. H. Pitkowsky, *Structural aspects of the System/360 Model 85 Part I: General organization* **7**, 1, 2 (1968).

Cook, R. L., F. L. Rawson III, J. A. Tunkel, and R. L. Williams, *Writing an Operating System/2 application* **27**, 2, 134 (1988); *G321-5315.*

Coons, S. A., see Ahuja, D. V., **7**, 3/4, 188 (1968).

Cooper, D. W., see Bitonti, F., **4**, 3, 200 (1965).

Cooper, J. C., *A capacity planning methodology* **19**, 1, 28 (1980); *G321-5114.*

Cordell, R. Q., II, M. Misra, and R. F. Wolfe, *Advanced Interactive Executive program development environment* **26**, 4, 361 (1987); *G321-5302.*

Cornish, R. E., see Gagliano, F. W., **7**, 3/4, 281 (1968).

Corr, F. P., and D. H. Neal, *SNA and emerging international standards* **18**, 2, 244 (1979); *G321-5095.*

Coryell, D. A., see Ruggiero, J. F., **8**, 2, 118 (1969).

Cowlishaw, M. F., *The design of the REXX language* **23**, 4, 326 (1984); *G321-5228.*

Cowlishaw, M. F., see Chess, D. M., **26**, 1, 138 (1987).

Crabtree, R. P., *Job networking* **17**, 3, 206 (1978); *G321-5071.*

Creasy, R. J., *Interactive Graphics in Data Processing Neutron cross-section evaluation* **7**, 3/4, 355 (1968).

Crowder, H., see Chen, R., **17**, 1, 82 (1978).

Cruise, A. V., see Milliken, K. R., **25**, 2, 159 (1986).

Crus, R. A., *Data recovery in IBM Database 2* **23**, 2, 178 (1984); *G321-5217.*

Cudney, L. F., see Blakeney, G. R., **6**, 2, 80 (1967).

Cullum, P. G., *The transmission subsystem in Systems Network Architecture* **15**, 1, 24 (1976); *G321-5026.*

Daly, T. P., see Bozman, G., **23**, 1, 44 (1984).

Daniel, C. J., see Heyne, G. F., **16**, 4, 344 (1977).

Dankowski, T. P., see Appel, A., **7**, 3/4, 176 (1968).

Darga, K., *On-line inquiry under a small-system operating system* **9**, 1, 2 (1970).

Dash, J. R., and R. N. Ojala, *IBM Database 2 in an Information Management System environment* **23**, 2, 165 (1984); *G321-5216.*

Dassinger, R. A., see Judisch, J. M., **20**, 2, 172 (1981).

Dave, J. V., see Halpern, P., **27**, 4, 561 (1988).

Davies, C. T., Jr., *Data processing spheres of control* **17**, 2, 179 (1978); *G321-5070.*

Davies, P. M., *Readings in microprogramming* **11**, 1, 16 (1972).

Davis, G. B., *Strategies for information requirements determination* **21**, 1, 4 (1982); *G321-5159.*

Davison, G., see Delgalvis, I., **3**, 1, 2 (1964).

Davison, R. W., see Buchwald, L. A., **27**, 3, 315 (1988).

Day, C. W., and L. L. Zimmerman, *Interactive Graphics in Data Processing Implementation and usage* **7**, 3/4, 373 (1968).

Day, W. H. E., *Compiler assignment of data items to registers* **9**, 4, 281 (1970).

Dean, M., *How a computer should talk to people* **21**, 4, 424 (1982); *G321-5175.*

Dean, W. M., see Bethke, F. J., **20**, 3, 306 (1981).

Deaton, G. A., Jr., and R. O. Hippert, Jr., *X.25 and related recommendations in IBM products* **22**, 1/2, 11 (1983); *G321-5180.*

Deaton, G. A., Jr., see Sundstrom, R. J., **26**, 1, 13 (1987).

deBry, R. K., and B. G. Platte, *Advanced Function Printing: A tutorial* **27**, 2, 219 (1988); *G321-5321.*

deBry, R. K., B. G. Platte, C. L. Berinato, and J. W. Marlin, *Architectures of Advanced Function Printing* **27**, 2, 234 (1988); *G321-5322.*

de Freitas, S. L., and P. J. Lavelle, *A method for the time analysis of programs* **17**, 1, 26 (1978); *G321-5062.*

Deitch, M., *Analytic queuing model for CICS capacity planning* **21**, 4, 454 (1982); *G321-5176.*

Delgalvis, I., and G. Davison, *Storage requirements for a data exchange* **3**, 1, 2 (1964).

Delgalvis, I., see Bricault, J. P., **5**, 3, 148 (1966).

DeMatteis, J. J., *An economic lot-sizing technique Part I: The part-period algorithm* **7**, 1, 30 (1968).

Demers, R. A., *Distributed files for SAA* **27**, 3, 348 (1988); *G321-5330.*

DeSousa, M. R., *Electronic information interchange in an office environment* **20**, 1, 4 (1981); *G321-5138.*

Devereaux, J. A., *An application-oriented multiprocessing system Part III: Control program features* **6**, 2, 95 (1967).

Devlin, B. A., and P. T. Murphy, *An architecture for a business and information system* **27**, 1, 60 (1988); *G321-5311.*

Dewell, M. E., see Haynes, W. K., **27**, 3, 384 (1988).

Dillon, T. J., see Jacobs, J. H., **9**, 2, 145 (1970).

Dimsdale, B., *Computer construction of minimal project networks* **2**, 1, 24 (1963).

Dimsdale, B., and H. P. Flatt, *Project evaluation and selection* **2**, 3/4, 200 (1963).

Dimsdale, B., and H. M. Markowitz, *A description of the SIMSCRIPT language* **3**, 1, 57 (1964).

Dixon, R. C., N. C. Strole, and J. D. Markov, *A token-ring network for local data communications* **22**, 1/2, 47 (1983); *G321-5182.*

Doherty, W. J., and R. P. Kelisky, *Managing VM/CMS systems for user effectiveness* **18**, 1, 143 (1979); *G321-5090.*

Doherty, W. J., and W. G. Pope, *Computing as a tool for human augmentation* **25**, 3/4, 306 (1986); *G321-5277.*

Donnan, R. A., and J. R. Kersey, *Synchronous data link control: A perspective* **13**, 2, 140 (1974).

Donovan, J. J., and S. E. Madnick, *Hierarchical approach to computer system integrity* **14**, 2, 188 (1975); *G321-5010.*

Dorrance, R. T., *Design of an integrated programming and operating system Part V: The system's COBOL compiler* **2**, 3/4, 322 (1963).

Dougherty, R. L., see Appel, A., **7**, 3/4, 176 (1968).

Dougherty, R. L., see Chen, F. C., **7**, 3/4, 257 (1968).

Drummond, M. E., Jr., *A perspective on system performance evaluation* **8**, 4, 252 (1969).

Duke, K. A., and W. A. Wall, *A professional graphics controller* **24**, 1, 14 (1985); *G321-5234.*

Duke, M. O., *Testing in a complex systems environment* **14**, 4, 353 (1975); *G321-5021.*

Dunbar, C. W., see Berk, M. A., **14**, 1, 64 (1975).

Dunfee, W. P., J. D. McGehe, R. C. Rauf, and K. O. Shipp, *Designing SAA applications and user interfaces* **27**, 3, 325 (1988); *G321-5329.*

Durbeck, R. C., see Mayadas, A. F., **25**, 3/4, 399 (1986).

Dyer, M., *The management of software engineering Part IV: Software development practices* **19**, 4, 451 (1980); *G321-5133.*

Dzielinski, B. P., *Fabrication and assembly operations Part IV: Linear programming in production planning* **4**, 2, 122 (1965).

Dzielinski, B. P., *A guide to financial planning tools and techniques* **12**, 2, 126 (1973).

Eade, D. J., P. Homan, and J. H. Jones, *CICS/VS and its role in Systems Network Architecture* **16**, 3, 258 (1977); *G321-5054.*

Effelsberg, W., see Fleischmann, A., **26**, 3, 255 (1987).

Efron, R., and G. Gordon, *A general purpose digital simulator and examples of its application Part I: Description of the simulator* **3**, 1, 22 (1964).

Ehrsam, W. F., S. M. Matyas, C. H. Meyer, and W. L. Tuchman, *A cryptographic key management scheme for implementing the Data Encryption Standard* **17**, 2, 106 (1978); *G321-5066.*

Eibsen, R. L., see Brown, D. T., **11**, 3, 186 (1972).

Eickhorn, C. R., see Blakeney, G. R., **6**, 2, 80 (1967).

Eisenbies, J. L., *Conventions for digital data communication link design* **6**, 4, 267 (1967).

Elson, M., and S. T. Rake, *Code-generation technique for large-language compilers* **9**, 3, 166 (1970).

Engel, G. H., J. Groppuso, R. A. Lowenstein, and W. G. Traub, *An office communications system* **18**, 3, 402 (1979); *G321-5102.*

Ennis, R. L., see Milliken, K. R., **25**, 2, 159 (1986).

Esau, L. R., and K. C. Williams, *On teleprocessing system design Part II: A method for approximating the optimal network* **5**, 3, 142 (1966).

Eswaran, K. P., see Blasgen, M. W., **16**, 4, 363 (1977).

Evangelisti, C. J., see Belady, L. A., **10**, 2, 143 (1971).

Evangelisti, C. J., see Belady, L. A., **19**, 4, 542 (1980).

Fagan, M. E., *Design and code inspections to reduce errors in program development* **15**, 3, 182 (1976); *G321-5033.*

Falkoff, A. D., K. E. Iverson, and E. H. Sussenguth, *A formal description of SYSTEM/360* **3**, 2, 198 (1964).

Farrell, E. J., *Visual interpretation of complex data* **26**, 2, 174 (1987); *G321-5293.*

Fehder, P. L., see Senko, M. E., **12**, 1, 30 (1973).

Fehder, P. L., see Senko, M. E., **12**, 1, 45 (1973).

Fehder, P. L., see Senko, M. E., **12**, 1, 64 (1973).

Felix, C. P., see Walston, C. E., **16**, 1, 54 (1977).

Fennel, J. W., Jr., and B. D. Gobioff, *A Satellite Communications Controller* **22**, 1/2, 81 (1983); *G321-5184.*

Ferdinand, A. E., *An analysis of the machine interference model* **10**, 2, 129 (1971).

Fernandez, E. B., see Wood, C., **19**, 2, 229 (1980).

Finkel, A. J., see Milliken, K. R., **25**, 2, 159 (1986).

Fisher, J. D., see Green, P. E., Jr., **26**, 4, 414 (1987).

Fitsos, G. P., see Christensen, K., **20**, 4, 372 (1981).

Fitzgerald, A. K., and B. F. Goodrich, *Data Management for the Distributed Processing Programming Executive (DPPX)* **18**, 4, 547 (1979); *G321-5109.*

Flagg, R. L., see Burns, N. A., **25**, 2, 236 (1986).

Flaherty, M. J., *Programming process productivity measurement system for System/370* **24**, 2, 168 (1985); *G321-5246.*

Flatt, H. P., see Dimsdale, B., **2**, 3/4, 200 (1963).

Fleischmann, A., S. T. Chin, and W. Effelsberg, *Specification and implementation of an ISO session layer* **26**, 3, 255 (1987); *G321-5297.*

Ford, J. B., *Enhanced problem determination capability for teleprocessing* **17**, 3, 276 (1978); *G321-5075.*

Frayne, D. N., see Bitonti, F., **4**, 3, 200 (1965).

Frazer, W. D., *Potential technology implications for computers and telecommunications in the 1980s* **18**, 2, 333 (1979); *G321-5099.*

Fredericksen, D. H., *Describing data in computer networks* **12**, 3, 257 (1973).

Freeman, D. N., *Macro language design for SYSTEM/360* **5**, 2, 62 (1966).

Freeman, D. N., see Bender, G., **6**, 1, 2 (1967).

Friedman, H. P., and G. Waldbaum, *Evaluating system changes under uncontrolled workloads: a case study* **14**, 4, 340 (1975); *G321-5020.*

Friedman, T. D., *The authorization problem in shared files* **9**, 4, 258 (1970).

Fromm, J. E., and D. E. Schreiber, *System aspects of large-problem computation and display* **11**, 1, 41 (1972).

Frosch, P. S., see Green, P. E., Jr., **26**, 4, 414 (1987).

Gafka, G., see Ahmadi, M., **27**, 1, 32 (1988).

Gagliano, F. W., H. W. Thombs, and R. E. Cornish, *Interactive Graphics in Data Processing A conversational display capability* **7**, 3/4, 281 (1968).

Galati, G. L., see Anderson, H. A., Jr., **14**, 3, 246 (1975).

Galli, E. J., and H. Yamada, *An automatic dictionary and the verification of machine-readable text* **6**, 3, 192 (1967).

Ganek, A. G., see Wheeler, E. F., **27**, 3, 250 (1988).

Gardner, P. C., Jr., *A system for the automated office environment* **20**, 3, 321 (1981); *G321-5152.*

Gardy, M., see Hagamen, W. D., **25**, 2, 207 (1986).

Garrigues, R. D., *An application of network management at a large computing service* **22**, 1/2, 143 (1983); *G321-5188.*

Gay, T. W., Jr., *On teleprocessing system design Part V: A technique for estimating channel interference* **5**, 3, 171 (1966).

Gazis, D. C., see Black, B. C., **10**, 3, 217 (1971).

Gecsei, J., see Mattson, R. L., **9**, 2, 78 (1970).

Gecsei, J., and J. A. Lukes, *A model for the evaluation of storage hierarchies* **13**, 2, 163 (1974).

George, F. D., and G. E. Young, *SNA flow control: Architecture and implementation* **21**, 2, 179 (1982); *G321-5166.*

Gerald, J., see Hatfield, D. J., **10**, 3, 168 (1971).

Gerson, G., *On modifying the 1620 ADD table* **1**, 1, 82 (1962).

Ghanem, S. B., *Computing center optimization by a pricing-priority policy* **14**, 3, 272 (1975); *G321-5015.*

Ghiotti, M. M., *A single-system interface using the IBM 3270-PC* **24**, 3/4, 236 (1985); *G321-5251.*

Gibson, D. H., see Conti, C. J., **7**, 1, 2 (1968).

Gibson, D. H., D. W. Rain, and H. F. Walsh, *Engineering and scientific processing on the IBM 3090* **25**, 1, 36 (1986); *G321-5260.*

Gibson, W. B., see Baldwin, F. R., **1**, 1, 64 (1962).

Giddings, G. M., see Weller, D. L., **19**, 3, 314 (1980).

Giesin, F. W., Jr., see Bryant, P., **15**, 2, 124 (1976).

Ginzberg, M. G., *Notes on testing real-time system programs* **4**, 1, 58 (1965).

Gladney, H. M., E. L. Worley, and J. J. Myers, *An access control mechanism for computing resources* **14**, 3, 212 (1975); *G321-5011.*

Gladney, H. M., D. L. Johnson, and R. L. Stone, *Computer installation accounting* **14**, 4, 314 (1975); *G321-5019.*

Gladney, H. M., see Raimondi, D. L., **15**, 1, 81 (1976).

Gladney, H. M., *Administrative control of computing service* **17**, 2, 151 (1978); *G321-5069.*

Gobioff, B. D., see Fennel, J. W., Jr., **22**, 1/2, 81 (1983).

Godard, D. N., see Green, P. E., Jr., **26**, 1, 37 (1987).

Goldberg, R., *Software engineering: An emerging discipline* **25**, 3/4, 334 (1986); *G321-5279.*

Goldstein, B. C., A. R. Heller, F. H. Moss, and I. Wladawsky-Berger, *Directions in cooperative processing between workstations and hosts* **23**, 3, 236 (1984); *G321-5221.*

Goldstein, B. C., and J. M. Jaffe, *Data communications: The implications of communication systems for protocol design* **26**, 1, 122 (1987); *G321-5290.*

Gomory, R. E., *The trim problem* **1**, 1, 77 (1962).

Goodfellow, M. J., see Chamberlin, D. D., **21**, 3, 250 (1982).

Goodman, A. S., see Lewis, P. A. W., **8**, 2, 136 (1969).

Goodrich, B. F., see Fitzgerald, A. K., **18**, 4, 547 (1979).

Gordon, G., *A general purpose systems simulator* **1**, 1, 18 (1962).

Gordon, G., see Blake, K., **3**, 1, 14 (1964).

Gordon, G., see Efron, R., **3**, 1, 22 (1964).

Gordon, R. D., *The Modular Application Customizing System* **19**, 4, 521 (1980); *G321-5136.*

Gordon, R. J., *Financial modeling on small systems* **12**, 2, 161 (1973).

Gorenstein, S., *Fabrication and assembly operations Part VI: Parameter values for sequencing control* **4**, 3, 241 (1965).

Gorenstein, S., *Programming for economic lot-sizes with precedences between items* **10**, 3, 232 (1971).

Gottschalk, K. D., *The System Usability Process for Network Management Products* **25**, 1, 83 (1986); *G321-5263.*

Gould, J. D., and S. J. Boies, *Speech filing–An office system for principals* **23**, 1, 65 (1984); *G321-5210.*

Gould, R. L., *GPSS/360–an improved general purpose simulator* **8**, 1, 16 (1969).

Gravina, C. M., *National Westminster Bank mass storage archiving* **17**, 4, 344 (1978); *G321-5079.*

Gray, J. N., see Blasgen, M. W., **20**, 1, 41 (1981).

Gray, J. P., and T. B. McNeill, *SNA multiple-system networking* **18**, 2, 263 (1979); *G321-5096.*

Gray, J. P., P. J. Hansen, P. Homan, M. A. Lerner, and M. Pozefsky, *Advanced program-to-program communication in SNA* **22**, 4, 298 (1983); *G321-5197.*

Green, P. E., *An introduction to network architectures and protocols* **18**, 2, 202 (1979); *G321-5093.*

Green, P. E., Jr., and D. N. Godard, *Prospects and design choices for integrated private networks* **26**, 1, 37 (1987); *G321-5286.*

Green, P. E., Jr., R. J. Chappuis, J. D. Fisher, P. S. Frosch, and C. E. Wood, *A perspective on Advanced Peer-to-Peer Networking* **26**, 4, 414 (1987); *G321-5305.*

Greenberg, J., see Tetzlaff, W. H., **26**, 2, 215 (1987).

Griffee, A. W., and C. A. Casey, *An introduction to typographic fonts and digital font resources* **27**, 2, 206 (1988); *G321-5320.*

Griffin, J. S., Jr., J. H. King, Jr., and C. J. Tunis, *A pattern identification system using linear decision functions* **2**, 3/4, 248 (1963).

Griffith, J. E., *An intrinsically addressed processing system* **2**, 3/4, 182 (1963).

Griffiths, P. D., see Roberts, M. L., **24**, 1, 49 (1985).

Groppuso, J., see Engel, G. H., **18**, 3, 402 (1979).

Grossman, C. P., *Cache-DASD storage design for improving system performance* **24**, 3/4, 316 (1985); *G321-5257.*

Grover, G. A., and K. Bharath-Kumar, *Windows in the sky–Flow control in SNA networks with satellite links* **22**, 4, 451 (1983); *G321-5205.*

Grover, G. A., see Sultan, R. A., **26**, 4, 429 (1987).

Gruhn, A. M., and A. C. Hohl, *A research perspective on computer-assisted office work* **18**, 3, 432 (1979); *G321-5103.*

Guido, A. A., see Cole, H., **16**, 1, 41 (1977).

Gumaer, R. A., see Teng, J. Z., **23**, 2, 211 (1984).

Haderle, D. J., and R. D. Jackson, *IBM Database 2 overview* **23**, 2, 112 (1984); *G321-5213.*

Hagamen, W. D., D. J. Linden, H. S. Long, and J. C. Weber, *Encoding verbal information as unique numbers* **11**, 4, 278 (1972).

Hagamen, W. D., D. J. Linden, K. F. Mai, S. M. Newell, and J. C. Weber, *A program generator* **14**, 2, 102 (1975); *G321-5007.*

Hagamen, W. D., and M. Gardy, *The numeric representation of knowledge and logic–Two artificial intelligence applications in medical education* **25**, 2, 207 (1986); *G321-5272.*

Hahn, S. G., and E. V. Hankam, *Kernel analysis of elliptic partial differential equations* **5**, 4, 248 (1966).

Haines, L. H., *Serial compilation and the 1401 FORTRAN compiler* **4**, 1, 73 (1965).

Hall, L. V., see Branin, F. H., Jr., **4**, 1, 2 (1965).

Hallman, H. K., see Carpenter, M. B., **24**, 2, 121 (1985).

Halpern, P., and J. W. Rettberg, *User-definable software applied to a real-time ambient air quality monitoring system* **20**, 1, 86 (1981); *G321-5142.*

Halpern, P., S. M. Roberts, and L. Lopez, *An incidence-matrix-driven panel system for the IBM PC* **26**, 2, 201 (1987); *G321-5294.*

Halpern, P., and J. V. Dave, *Interactive computations and display of characteristics of the radiation scattered by a sphere: A demonstration for PS/2 Model 80* **27**, 4, 561 (1988); *G321-5343.*

Halsey, J. R., L. E. Hardy, and L. F. Powning, *Public data networks: Their evolution, interfaces, and status* **18**, 2, 223 (1979); *G321-5094.*

Hammond, K. R., and M. R. Zimowski, *TSO Attach: A multipurpose communication channel to IBM Database 2* **23**, 2, 151 (1984); *G321-5215.*

Hammond, L. W., *Management considerations for an Information Center* **21**, 2, 131 (1982); *G321-5164.*

Handelman, S. W., *A high-resolution computer graphics system* **19**, 3, 356 (1980); *G321-5130.*

Hanford, K. V., *Automatic generation of test cases* **9**, 4, 242 (1970).

Hankam, E. V., see Hahn, S. G., **5**, 4, 248 (1966).

Hansen, H. H., see Bitonti, F., **4**, 3, 200 (1965).

Hansen, P. J., see Gray, J. P., **22**, 4, 298 (1983).

Hanssmann, F., W. Kistler, and H. Schultz, *Modeling for computing center planning* **10**, 4, 305 (1971).

Harding, J. T., see Radice, R. A., **24**, 2, 91 (1985).

Hardy, L. E., see Halsey, J. R., **18**, 2, 223 (1979).

Harrison, E. S., *Distributed processing communications software support for operation within an SNA network* **19**, 2, 192 (1980); *G321-5122.*

Harrison, E. S., and E. J. Schmitt, *The structure of System/88, a fault-tolerant computer* **26**, 3, 293 (1987); *G321-5299.*

Hartmann, T. C., see Hendricks, E. C., **18**, 1, 111 (1979).

Hassitt, A., and L. E. Lyon, *An APL emulator on System/370* **15**, 4, 358 (1976); *G321-5041.*

Hatfield, D. J., and J. Gerald, *Program restructuring for virtual memory* **10**, 3, 168 (1971).

Hatfield, D. J., see Parmelee, R. P., **11**, 2, 99 (1972).

Hauth, C., *Turnaround time for messages of differing priorities* **7**, 2, 103 (1968).

Havender, J. W., *Avoiding deadlock in multitasking systems* **7**, 2, 74 (1968).

Hayes, R. M., see Bryant, P., **15**, 2, 124 (1976).

Haynes, W. K., M. E. Dewell, and P. J. Herman, *The Cross System Product application generator: An evolution* **27**, 3, 384 (1988); *G321-5333.*

Hedberg, R., *Design of an integrated programming and operating system Part III: The expanded function of the loader* **2**, 3/4, 298 (1963).

Heidorn, G. E., K. Jensen, L. A. Miller, R. J. Byrd, and M. S. Chodorow, *The EPISTLE text-critiquing system* **21**, 3, 305 (1982); *G321-5171.*

Hein, K. P., *Information System Model and Architecture Generator* **24**, 3/4, 213 (1985); *G321-5250.*

Heising, W. P., *Note on random addressing techniques* **2**, 2, 112 (1963).

Helander, G. A., *Improving system usability for business professionals* **20**, 3, 294 (1981); *G321-5150.*

Held, M., and R. M. Karp, *The construction of discrete dynamic programming algorithms* **4**, 2, 136 (1965).

Heller, A. R., see Goldstein, B. C., **23**, 3, 236 (1984).

Hellerman, H., *Some principles of time-sharing scheduler strategies* **8**, 2, 94 (1969).

Hellerstein, J. L., see Milliken, K. R., **25**, 2, 159 (1986).

Hendricks, E. C., and T. C. Hartmann, *Evolution of a virtual machine subsystem* **18**, 1, 111 (1979); *G321-5089.*

Henn, H. H., see Bleher, J. H., **19**, 3, 382 (1980).

Henry, G. G., *IBM small-system architecture and design–Past, present, and future* **25**, 3/4, 321 (1986); *G321-5278.*

Henry, W. R., *Hierarchical structure for data management* **8**, 1, 2 (1969).

Herman, P. J., see Haynes, W. K., **27**, 3, 384 (1988).

Herscovitch, H., and T. Schneider, *GPSS III–An expanded general purpose simulator* **4**, 3, 174 (1965).

Hertel, H. F., see Stanley, W. I., **7**, 2, 85 (1968).

Hess, M. L., see Benjamin, J. H., **22**, 4, 344 (1983).

Hess, M. L., see Sundstrom, R. J., **26**, 1, 13 (1987).

Hester, P. D., see Simpson, R. O., **26**, 4, 346 (1987).

Hevner, A. R., see Mills, H. D., **26**, 4, 395 (1987).

Heyne, G. F., and C. J. Daniel, *Design techniques for a user controlled DB/DC system* **16**, 4, 344 (1977); *G321-5057.*

Hinsberg, W. D., see Mayadas, A. F., **25**, 3/4, 399 (1986).

Hippert, R. O., Jr., L. R. Palounek, J. Provetero, and R. O. Skatrud, *Reliability, availability, and serviceability design considerations for the Supermarket and Retail Store Systems* **14**, 1, 81 (1975); *G321-5006.*

Hippert, R. O., Jr., see Deaton, G. A., Jr., **22**, 1/2, 11 (1983).

Hirsch, R. S., *Procedures of the Human Factors Center at San Jose* **20**, 2, 123 (1981); *G321-5144.*

Hobgood, W. S., *Evaluation of an interactive-batch system network* **11**, 1, 2 (1972).

Hobgood, W. S., *The role of the Network Control Program in Systems Network Architecture* **15**, 1, 39 (1976); *G321-5025.*

Hobson, G. C., see Berk, M. A., **14**, 1, 64 (1975).

Hochweller, G., see Raimondi, D. L., **15**, 1, 81 (1976).

Hodil, E. D., C. W. Butler, and G. L. Richardson, *Knowledge-based systems in the commercial environment* **25**, 2, 147 (1986); *G321-5268.*

Hoffnagle, G. F., and W. E. Beregi, *Automating the software development process* **24**, 2, 102 (1985); *G321-5242.*

Hofmann, C. A., see Bird, R. A., **19**, 1, 140 (1980).

Hohl, A. C., see Gruhn, A. M., **18**, 3, 432 (1979).

Holley, L. H., R. P. Parmelee, C. A. Salisbury, and D. N. Saul, *VM/370 asymmetric multiprocessing* **18**, 1, 47 (1979); *G321-5086.*

Homan, P., see Eade, D. J., **16**, 3, 258 (1977).

Homan, P., see Gray, J. P., **22**, 4, 298 (1983).

Honjou, N., see Sakaki, M., **27**, 4, 528 (1988).

Hopkins, M. E., *A perspective on the 801/Reduced Instruction Set Computer* **26**, 1, 107 (1987); *G321-5289.*

Housel, B. C., and C. J. Scopinich, *SNA Distribution Services* **22**, 4, 319 (1983); *G321-5198.*

Howard, J. A., see Baily, J. H., **13**, 3, 253 (1974).

Howry, S. K., see McGee, W. C., **7**, 3/4, 342 (1968).

Hubbard, G. U., see Raver, N., **16**, 3, 287 (1977).

Humphrey, W. S., *The IBM large-systems software development process: Objectives and direction* **24**, 2, 76 (1985); *G321-5239.*

Hunsinger, L. E., see Stacy, E. W., **8**, 3, 220 (1969).

Huth, H. T., see McEnroe, P. V., **14**, 1, 3 (1975).

Iacobucci, E. E., see Weingarten, R. A., **22**, 4, 387 (1983).

Ish-Shalom, J., see Korein, J. U., **26**, 1, 55 (1987).

Iverson, K. E., *Programming notation in systems design* **2**, 2, 117 (1963).

Iverson, K. E., see Falkoff, A. D., **3**, 2, 198 (1964).

Iwaskiw, C. T., see Burns, N. A., **25**, 2, 236 (1986).

Jackson, R. D., see Haderle, D. J., **23**, 2, 112 (1984).

Jacobs, J. H., and T. J. Dillon, *Interactive Saturn flight program simulator* **9**, 2, 145 (1970).

Jacobsen, J. D., *Interactive Graphics in Data Processing Geometric relationships for retrieval of geographic information* **7**, 3/4, 331 (1968).

Jaffe, J. F., see Auslander, M. A., **12**, 4, 368 (1973).

Jaffe, J. M., F. H. Moss, and R. A. Weingarten, *SNA routing: Past, present, and possible future* **22**, 4, 417 (1983); *G321-5203.*

Jaffe, J. M., see Goldstein, B. C., **26**, 1, 122 (1987).

Jensen, K., see Heidorn, G. E., **21**, 3, 305 (1982).

Jensen, R. M., *A formal approach for communication between logically isolated virtual machines* **18**, 1, 71 (1979); *G321-5087.*

Jephson, J. S., R. P. McQuarrie, and R. E. Vogelsberg, *A three-value computer design verification system* **8**, 3, 178 (1969).

Johnson, C. I., *Interactive Graphics in Data Processing Principles of interactive systems* **7**, 3/4, 147 (1968).

Johnson, D. L., see Gladney, H. M., **14**, 4, 314 (1975).

Johnson, E. L., see Chen, R., **17**, 1, 82 (1978).

Johnson, S. K., see Percival, L. C., **25**, 1, 92 (1986).

Jones, A. L., see Niehoff, W. H., **19**, 3, 367 (1980).

Jones, A. U., *A user-oriented data-base retrieval system* **16**, 1, 4 (1977); *G321-5042.*

Jones, C. L., *A process-integrated approach to defect prevention* **24**, 2, 150 (1985); *G321-5245.*

Jones, H. C., see Anastassiou, D., **22**, 1/2, 97 (1983).

Jones, J. H., see Eade, D. J., **16**, 3, 258 (1977).

Jones, T. C., *Measuring programming quality and productivity* **17**, 1, 39 (1978); *G321-5063.*

Joslin, P. H., *System Productivity Facility* **20**, 4, 388 (1981); *G321-5155.*

Judisch, J. M., B. A. Rupp, and R. A. Dassinger, *Effects of manual style on performance in education and machine maintenance* **20**, 2, 172 (1981); *G321-5145.*

Kahn, S., *An overview of three relational data base products* **23**, 2, 100 (1984); *G321-5212.*

Kaiser, P. H., see Bethke, F. J., **20**, 3, 306 (1981).

Kamel, A., M. Kindelan, and P. Sguazzero, *Seismic computations on the IBM 3090 Vector Multiprocessor* **27**, 4, 510 (1988); *G321-5340.*

Kanyuh, D., *An integrated network management product* **27**, 1, 45 (1988); *G321-5310.*

Karp, D. P., see McKay, D. B., **12**, 1, 94 (1973).

Karp, R. M., see Held, M., **4**, 2, 136 (1965).

Kasson, J. M., *An advanced voice/data telephone switching system* **25**, 3/4, 380 (1986); *G321-5282.*

Katz, S., L. J. Risman, and M. Rodeh, *A system for constructing linear programming models* **19**, 4, 505 (1980); *G321-5135.*

Keefe, D. D., *Hierarchical control programs for systems evaluation* **7**, 2, 123 (1968).

Keehn, D. G., and J. O. Lacy, *VSAM data set design parameters* **13**, 3, 186 (1974).

Keeley, J. F., *An application-oriented multiprocessing system Part I: Introduction* **6**, 2, 78 (1967).

Kelisky, R. P., see Doherty, W. J., **18**, 1, 143 (1979).

Kermani, P., see Sultan, R. A., **26**, 4, 429 (1987).

Kersey, J. R., see Donnan, R. A., **13**, 2, 140 (1974).

Kiely, S. C., *An operating system for distributed processing–DPPX* **18**, 4, 507 (1979); *G321-5107.*

Kindelan, M., see Kamel, A., **27**, 4, 510 (1988).

King, G. M., see Singh, Y., **25**, 1, 20 (1986).

King, J. C., see Chamberlin, D. D., **21**, 3, 250 (1982).

King, J. H., Jr., see Griffin, J. S., Jr., **2**, 3/4, 248 (1963).

King, W. F., see Blasgen, M. W., **20**, 1, 41 (1981).

Kingston, P. L., *Concepts of financial models* **12**, 2, 113 (1973).

Kistler, W., see Hanssmann, F., **10**, 4, 305 (1971).

Klein, D. A., see Milliken, K. R., **25**, 2, 159 (1986).

Kobayashi, Y., see Sy, K. K., **26**, 2, 157 (1987).

Kogan, M. S., and F. L. Rawson III, *The design of Operating System/2* **27**, 2, 90 (1988); *G321-5312.*

Kolsky, H. G., *Problem formulation using APL* **8**, 3, 204 (1969).

Konheim, A. G., M. H. Mack, R. K. McNeill, B. Tuckerman, and G. Waldbaum, *The IPS cryptographic programs* **19**, 2, 253 (1980); *G321-5125.*

Korein, J. U., and J. Ish-Shalom, *Robotics* **26**, 1, 55 (1987); *G321-5287.*

Korn, P. A., J. P. McAdaragh, and C. L. Tondo, *Expanded personal computing power and capability* **24**, 1, 26 (1985); *G321-5235.*

Kossack, C. F., *Optimum response analysis* **2**, 1, 49 (1963).

Kossack, C. F., *Statistical classification techniques* **2**, 2, 136 (1963).

Koster, L. R., see McCrossin, J. M., **17**, 3, 260 (1978).

Koved, L., and G. Waldbaum, *Improving availability of software subsystems through on-line error detection* **25**, 1, 105 (1986); *G321-5265.*

Kozuh, F. T., D. L. Livingston, and T. C. Spillman, *System/370 capability in a desktop computer* **23**, 3, 245 (1984); *G321-5222.*

Kraemer, W., *Performance investigations with a DOS/VS-based operating system model* **17**, 4, 409 (1978); *G321-5082.*

Krasnow, H. S., see Boyd, D. F., **2**, 1, 2 (1963).

Krasnow, H. S., see Boyd, D. F., **3**, 1, 51 (1964).

Kravitz, J. K., D. Lieber, F. H. Robbins, and J. M. Palermo, *Workstations and mainframe computers working together* **25**, 1, 116 (1986); *G321-5266.*

Kuki, H., and J. Ascoly, *FORTRAN extended-precision library* **10**, 1, 39 (1971).

Lacy, J. O., see Keehn, D. G., **13**, 3, 186 (1974).

Lafuente, J. M., *Interactive Graphics in Data Processing Cam design on a graphics console* **7**, 3/4, 365 (1968).

Laird, J. A., *The Power Profile–An installation management tool* **14**, 3, 264 (1975); *G321-5014.*

Lambert, G. N., *A comparative study of system response time on program developer productivity* **23**, 1, 36 (1984); *G321-5208.*

Lancto, D. C., and R. L. Rockefeller, *An application-oriented multiprocessing system Part IV: The operational error analysis program* **6**, 2, 103 (1967).

Lande, H. F., *Planning-data systems* **12**, 2, 145 (1973).

Larner, R., *Design of an integrated programming and operating system Part IV: The system's FORTRAN compiler* **2**, 3/4, 311 (1963).

Laurer, G. J., see Savir, D., **14**, 1, 16 (1975).

Lavelle, P. J., see de Freitas, S. L., **17**, 1, 26 (1978).

Leach, J. R., and R. D. Campenni, *A sidestream approach using a small processor as a tool for managing communication systems* **19**, 1, 120 (1980); *G321-5119.*

Leavenworth, B. M., see Archibald, J. L., **22**, 3, 170 (1983).

Lehman, M. M., see Shedler, G. S., **6**, 3, 142 (1967).

Lehman, M. M., see Belady, L. A., **15**, 3, 225 (1976).

Lennon, R. E., *Cryptography architecture for information security* **17**, 2, 138 (1978); *G321-5068.*

Lerner, M. A., see Gray, J. P., **22**, 4, 298 (1983).

Lewis, L. J., *Service levels: A concept for the user and the computer center* **15**, 4, 328 (1976); *G321-5040.*

Lewis, P. A. W., *A computer program for the statistical analysis of series of events* **5**, 4, 202 (1966).

Lewis, P. A. W., A. S. Goodman, and J. M. Miller, *A pseudo-random number generator for the System/360* **8**, 2, 136 (1969).

Lieber, D., see Kravitz, J. K., **25**, 1, 116 (1986).

Lind, R. A., see Seaman, P. H., **5**, 3, 158 (1966).

Linden, D. J., see Hagamen, W. D., **11**, 4, 278 (1972).

Linden, D. J., see Hagamen, W. D., **14**, 2, 102 (1975).

Lindsay, B. G., see Blasgen, M. W., **20**, 1, 41 (1981).

Linehan, M. H., see Mendelsohn, N., **22**, 1/2, 63 (1983).

Ling, D. T., see Matick, R. E., **23**, 3, 264 (1984).

Linger, R. C., *The management of software engineering Part III: Software design practices* **19**, 4, 432 (1980); *G321-5133.*

Linger, R. C., see Mills, H. D., **26**, 4, 395 (1987).

Liptay, J. S., *Structural aspects of the System/360 Model 85 Part II: The cache* **7**, 1, 15 (1968).

Livingston, D. L., see Kozuh, F. T., **23**, 3, 245 (1984).

Loeb, D. J., see Milliken, K. R., **25**, 2, 159 (1986).

Loewner, P. G., *Fabrication and assembly operations Part III: Matrix methods for processing configuration data* **4**, 2, 105 (1965).

Logan, D., see Clementi, E., **27**, 4, 475 (1988).

Long, H. S., see Hagamen, W. D., **11**, 4, 278 (1972).

Loosley, C. R., see Cheng, J. M., **23**, 2, 189 (1984).

Lopez, L., see Halpern, P., **26**, 2, 201 (1987).

Lorie, R. A., see Blasgen, M. W., **20**, 1, 41 (1981).

Lorin, H., *A guided bibliography to sorting* **10**, 3, 244 (1971).

Lorin, H., *Distributed processing: An assessment* **18**, 4, 582 (1979); *G321-5111.*

Lorin, H., *Systems architecture in transition–An overview* **25**, 3/4, 256 (1986); *G321-5275.*

Loucks, L. K., and C. H. Sauer, *Advanced Interactive Executive (AIX) operating system overview* **26**, 4, 326 (1987); *G321-5300.*

Low, D. W., see Brown, J. F., **12**, 2, 172 (1973).

Lowenstein, R. A., see Engel, G. H., **18**, 3, 402 (1979).

Lozupone, D. F., see Angeleri, P., **27**, 4, 541 (1988).

Luetje, J. H., *Interactive aeronautical charting* **9**, 3, 219 (1970).

Lukes, J. A., see Gecsei, J., **13**, 2, 163 (1974).

Lum, V. Y., D. M. Choy, and N. C. Shu, *OPAS: An office procedure automation system* **21**, 3, 327 (1982); *G321-5172.*

Lynch, H. W., and J. B. Page, *The OS/VS2 Release 2 System Resources Manager* **13**, 4, 274 (1974).

Lyon, L. E., see Hassitt, A., **15**, 4, 358 (1976).

Mack, M. H., see Konheim, A. G., **19**, 2, 253 (1980).

MacKinnon, R. A., *Advanced function extended with tightly-coupled multiprocessing* **13**, 1, 32 (1974).

MacKinnon, R. A., see Seawright, L. H., **18**, 1, 4 (1979).

MacKinnon, R. A., *The changing virtual machine environment: Interfaces to real hardware, virtual hardware, and other virtual machines* **18**, 1, 18 (1979); *G321-5085.*

Madnick, S. E., see Donovan, J. J., **14**, 2, 188 (1975).

Maerker, K., see Bleher, J. H., **19**, 3, 382 (1980).

Mai, K. F., see Hagamen, W. D., **14**, 2, 102 (1975).

Major, J. B., *Processor, I/O path, and DASD configuration capacity* **20**, 1, 63 (1981); *G321-5141.*

Malhotra, A., see Pazel, D. P., **22**, 3, 188 (1983).

Mancini, R. J., *Performance analysis for the Skylab terminal system* **13**, 2, 94 (1974).

Maranzana, F. E., *On the location of supply points to minimize transportation costs* **2**, 2, 129 (1963).

Margolin, B. H., R. P. Parmelee, and M. Schatzoff, *Analysis of free-storage algorithms* **10**, 4, 283 (1971).

Margopoulos, W. P., and R. J. Williams, *On teleprocessing system design Part I: Characteristic problems* **5**, 3, 134 (1966).

Markov, J. D., see Dixon, R. C., **22**, 1/2, 47 (1983).

Markowitz, H. M., see Dimsdale, B., **3**, 1, 57 (1964).

Markowitz, H. M., see Pazel, D. P., **22**, 3, 188 (1983).

Marks, B. L., *Design of a checkout compiler* **12**, 3, 315 (1973).

Markstein, P. W., see Attanasio, C. R., **15**, 1, 102 (1976).

Marlin, J. W., see deBry, R. K., **27**, 2, 234 (1988).

Martin, R. W., see Raimondi, D. L., **15**, 1, 81 (1976).

Maruyama, K., *Defining routing tables for SNA networks* **22**, 4, 435 (1983); *G321-5204.*

Masullo, M. J., see Milliken, K. R., **25**, 2, 159 (1986).

Matick, R. E., and D. T. Ling, *Architecture implications in the design of microprocessors* **23**, 3, 264 (1984); *G321-5224.*

Matick, R. E., *Impact of memory systems on computer architecture and system organization* **25**, 3/4, 274 (1986); *G321-5276.*

Mattson, R. L., J. Gecsei, D. R. Slutz, and I. L. Traiger, *Evaluation techniques for storage hierarchies* **9**, 2, 78 (1970).

Matyas, S. M., see Ehrsam, W. F., **17**, 2, 106 (1978).

Matyas, S. M., and C. H. Meyer, *Generation, distribution, and installation of cryptographic keys* **17**, 2, 126 (1978); *G321-5067.*

Maurer, M. E., *Full-screen testing of interactive applications* **22**, 3, 246 (1983); *G321-5194.*

Mayadas, A. F., R. C. Durbeck, W. D. Hinsberg, and J. M. McCrossin, *The evolution of printers and displays* **25**, 3/4, 399 (1986); *G321-5283.*

Mays, R. G., L. S. Orzech, W. A. Ciarfella, and R. W. Phillips, *PDM: A requirements methodology for software system enhancements* **24**, 2, 134 (1985); *G321-5244.*

McAdaragh, J. P., see Korn, P. A., **24**, 1, 26 (1985).

McClelland, P., see Banham, J. A., **11**, 4, 329 (1972).

McComb, J., and S. Schmidt, *Engineering and Scientific Subroutine Library for the IBM 3090 Vector Facility* **27**, 4, 404 (1988); *G321-5335.*

McCrossin, J. M., R. P. O'Hara, and L. R. Koster, *A time-sharing display terminal session manager* **17**, 3, 260 (1978); *G321-5074.*

McCrossin, J. M., see Mayadas, A. F., **25**, 3/4, 399 (1986).

McCue, G. M., *IBM's Santa Teresa Laboratory–Architectural design for program development* **17**, 1, 4 (1978); *G321-5061.*

McEnroe, P. V., H. T. Huth, E. A. Moore, and W. W. Morris III, *Overview of the Supermarket System and the Retail Store System* **14**, 1, 3 (1975); *G321-5001.*

McFadyen, J. H., *Systems Network Architecture: An overview* **15**, 1, 4 (1976); *G321-5024.*

McGee, W. C., *On dynamic program relocation* **4**, 3, 184 (1965).

McGee, W. C., H. R. Penafiel, and S. K. Howry, *Interactive Graphics in Data Processing Analysis and display of physics data* **7**, 3/4, 342 (1968).

McGee, W. C., *The information management system IMS/VS Part I: General structure and operation* **16**, 2, 84 (1977); *G321-5046.*

McGee, W. C., *The information management system IMS/VS Part II: Data base facilities* **16**, 2, 96 (1977); *G321-5047.*

McGee, W. C., *The information management system IMS/VS Part III: Batch processing facilities* **16**, 2, 123 (1977); *G321-5048.*

McGee, W. C., *The information management system IMS/VS Part IV: Data communication facilities* **16**, 2, 136 (1977); *G321-5049.*

McGee, W. C., *The information management system IMS/VS Part V: Transaction processing facilities* **16**, 2, 148 (1977); *G321-5050.*

McGehe, J. D., see Dunfee, W. P., **27**, 3, 325 (1988).

McGrath, M., *Virtual machine computing in an engineering environment* **11**, 2, 131 (1972).

McKay, D. B., and D. P. Karp, *Protocol for a computer network* **12**, 1, 94 (1973).

McKinstry, R. H., see Arnold, J. S., **13**, 1, 60 (1974).

McManigal, D. F., and D. A. Stevenson, *Architecture of the IBM 3277 Graphics Attachment* **19**, 3, 331 (1980); *G321-5128.*

McNeill, R. K., see Konheim, A. G., **19**, 2, 253 (1980).

McNeill, T. B., see Gray, J. P., **18**, 2, 263 (1979).

McPhee, W. S., *Operating system integrity in OS/VS2* **13**, 3, 230 (1974).

McQuarrie, R. P., see Jephson, J. S., **8**, 3, 178 (1969).

Mealy, G. H., *The functional structure of OS/360 Part I: Introductory survey* **5**, 1, 3 (1966).

Meggitt, J. E., *A character computer for high-level language interpretation* **3**, 1, 68 (1964).

Mehl, J. W., see Blasgen, M. W., **20**, 1, 41 (1981).

Mei, V., see Ricour, D. H., **6**, 1, 38 (1967).

Mendelsohn, N., M. H. Linehan, and W. J. Anzick, *Reflections on VM/Pass-Through: A facility for interactive networking* **22**, 1/2, 63 (1983); *G321-5183.*

Mendoza, A. G., *An economic lot-sizing technique Part II: Mathematical analysis of the part-period algorithm* **7**, 1, 39 (1968).

Metz, W. C., Jr., and D. Savir, *Store performance studies for the Supermarket System* **14**, 1, 46 (1975); *G321-5004.*

Meyer, C. H., see Ehrsam, W. F., **17**, 2, 106 (1978).

Meyer, C. H., see Matyas, S. M., **17**, 2, 126 (1978).

Meyer, R. A., and L. H. Seawright, *A virtual machine time-sharing system* **9**, 3, 199 (1970).

Miller, J. M., see Lewis, P. A. W., **8**, 2, 136 (1969).

Miller, L. A., *Natural language programming: Styles, strategies, and contrasts* **20**, 2, 184 (1981); *G321-5146.*

Miller, L. A., see Heidorn, G. E., **21**, 3, 305 (1982).

Milliken, K. R., A. V. Cruise, R. L. Ennis, A. J. Finkel, J. L. Hellerstein, D. J. Loeb, D. A. Klein, M. J. Masulio, H. M. Van Woerkom, and N. B. Waite, *YES/MVS and the automation of operations for large computer complexes* **25**, 2, 159 (1986); *G321-5269.*

Mills, H. D., *The management of software engineering Part I: Principles of software engineering* **19**, 4, 414 (1980); *G321-5133.*

Mills, H. D., R. C. Linger, and A. R. Hevner, *Box structured information systems* **26**, 4, 395 (1987); *G321-5304.*

Mishelevich, D. J., and D. Van Slyke, *Application development system: The software architecture of the IBM Health Care Support/DL/I-Patient Care System* **19**, 4, 478 (1980); *G321-5134.*

Misra, M., see Cordell, R. Q., II, **26**, 4, 361 (1987).

Misra, P. N., *Capacity analysis of the Mass Storage System* **20**, 3, 346 (1981); *G321-5153.*

Misra, P. N., *Software reliability analysis* **22**, 3, 262 (1983); *G321-5195.*

Mitchell, J. L., see Anastassiou, D., **22**, 1/2, 97 (1983).

Mizell, A. M., *Understanding device drivers in Operating System/2* **27**, 2, 170 (1988); *G321-5317.*

Montalbano, M., *Tables, flow charts, and program logic* **1**, 1, 51 (1962).

Montalbano, M., *High-speed calculation of the critical paths of large networks* **6**, 3, 163 (1967).

Moore, D. J., *Teletex–A worldwide link among office systems for electronic document exchange* **22**, 1/2, 30 (1983); *G321-5181.*

Moore, E. A., see McEnroe, P. V., **14**, 1, 3 (1975).

Moore, R. E., *Utilizing the SNA Alert in the management of multivendor networks* **27**, 1, 15 (1988); *G321-5308.*

Moore, R. S., *Evolution of a laboratory communication network* **18**, 2, 315 (1979); *G321-5098.*

Morris, W. W., III, see McEnroe, P. V., **14**, 1, 3 (1975).

Morrison, J. E., *User program performance in virtual storage systems* **12**, 3, 216 (1973).

Morrison, J. P., *Data Stream Linkage Mechanism* **17**, 4, 383 (1978); *G321-5081.*

Morse, S. P., see Baskin, H. B., **7**, 3/4, 218 (1968).

Moss, F. H., see Jaffe, J. M., **22**, 4, 417 (1983).

Moss, F. H., see Goldstein, B. C., **23**, 3, 236 (1984).

Mudie, M. W., and D. J. Schafer, *An information technology architecture for change* **24**, 3/4, 307 (1985); *G321-5256.*

Mueller, J. H., *Aspects of the Gemini real-time operating system* **6**, 3, 150 (1967).

Mugele, R. A., *A program for optimal control of nonlinear processes* **1**, 1, 2 (1962).

Munn, J. E., see Rose, D. B., **27**, 1, 3 (1988).

Munnis, P. E., see Radice, R. A., **24**, 2, 91 (1985).

Munoz, C. U., see Bird, D. L., **22**, 3, 229 (1983).

Murphy, P. T., see Devlin, B. A., **27**, 1, 60 (1988).

Murray, W. H., *Security considerations for personal computers* **23**, 3, 297 (1984); *G321-5226.*

Musgrave, J. F., *Experiments in computer-aided graphic expression* **17**, 3, 241 (1978); *G321-5073.*

Myers, G. J., see Stevens, W. P., **13**, 2, 115 (1974).

Myers, G. J., *Composite design facilities of six programming languages* **15**, 3, 212 (1976); *G321-5034.*

Myers, J. J., see Gladney, H. M., **14**, 3, 212 (1975).

Myers, J. J., see Considine, J. P., **16**, 4, 378 (1977).

Myre, W. W., see Chang, P. Y., **27**, 2, 105 (1988).

Nagler, H., *Recovery for computer switchover in a real-time system* **2**, 1, 76 (1963).

Nahouraii, E., *Direct-access device simulation* **13**, 1, 19 (1974).

Neal, A. S., and R. M. Simons, *Playback: A method for evaluating the usability of software and its documentation* **23**, 1, 82 (1984); *G321-5211.*

Neal, D. H., see Corr, F. P., **18**, 2, 244 (1979).

Neuhold, E. J., *The formal description of programming languages* **10**, 2, 86 (1971).

Newell, S. M., see Hagamen, W. D., **14**, 2, 102 (1975).

Newitt, J. W., and A. Odarchenko, *A structure for real-time Stenotype transcription* **9**, 1, 24 (1970).

Newman, P. S., *Towards an integrated development environment* **21**, 1, 81 (1982); *G321-5162.*

Newton, J., *Strategies for problem prevention* **24**, 3/4, 248 (1985); *G321-5252.*

Nguyen, H. C., A. Ockene, R. Revell, and W. J. Skwish, *The role of detailed simulation in capacity planning* **19**, 1, 81 (1980); *G321-5117.*

Niehoff, W. H., and A. L. Jones, *An APL approach to presentation graphics* **19**, 3, 367 (1980); *G321-5131.*

Noble, A. S., Jr., *Design of an integrated programming and operating system Part I: System considerations and the monitor* **2**, 2, 153 (1963).

Ockene, A., see Nguyen, H. C., **19**, 1, 81 (1980).

Odarchenko, A., see Newitt, J. W., **9**, 1, 24 (1970).

O'Hara, A. C., Jr., see Radice, R. A., **24**, 2, 79 (1985).

O'Hara, R. P., see McCrossin, J. M., **17**, 3, 260 (1978).

Ohms, B. G., *Computer processing of dates outside the twentieth century* **25**, 2, 244 (1986); *G321-5274.*

Ojala, R. N., see Dash, J. R., **23**, 2, 165 (1984).

Okaya, Y., see Cole, H., **2**, 3/4, 240 (1963).

Okaya, Y., *Interactive Graphics in Data Processing Interactive aspects of crystal structure analysis* **7**, 3/4, 322 (1968).

Ominsky, M., *Human factors in the development of a family of plant data communication terminals* **20**, 2, 216 (1981); *G321-5147.*

O'Neill, D., *The management of software engineering Part II: Software engineering program* **19**, 4, 421 (1980); *G321-5133.*

Orlicky, J. A., *Net change material requirements planning* **12**, 1, 2 (1973).

Ort, E., see Bethke, F. J., **20**, 3, 306 (1981).

Orzech, L. S., see Mays, R. G., **24**, 2, 134 (1985).

Padegs, A., *The structure of SYSTEM/360 Part IV: Channel design considerations* **3**, 2, 165 (1964).

Padegs, A., *Structural aspects of the System/360 Model 85 Part III: Extensions to floating-point architecture* **7**, 1, 22 (1968).

Page, J. B., see Lynch, H. W., **13**, 4, 274 (1974).

Palermo, F. P., see Weller, D. L., **19**, 3, 314 (1980).

Palermo, J. M., see Kravitz, J. K., **25**, 1, 116 (1986).

Palounek, L. R., see Hippert, R. O., Jr., **14**, 1, 81 (1975).

Parker, M. M., *Enterprise information analysis: Cost-benefit analysis and the data-managed system* **21**, 1, 108 (1982); *G321-5163.*

Parmelee, R. P., see Margolin, B. H., **10**, 4, 283 (1971).

Parmelee, R. P., T. I. Peterson, C. C. Tillman, and D. J. Hatfield, *Virtual storage and virtual machine concepts* **11**, 2, 99 (1972).

Parmelee, R. P., see Holley, L. H., **18**, 1, 47 (1979).

Parr, F. N., and R. E. Strom, *NIL: A high-level language for distributed systems programming* **22**, 1/2, 111 (1983); *G321-5186.*

Patterson, D. R., see Tetzlaff, W. H., **26**, 2, 215 (1987).

Pazel, D. P., A. Malhotra, and H. M. Markowitz, *The system architecture of EAS-E: An integrated programming and data base language* **22**, 3, 188 (1983); *G321-5190.*

Pearsall, R. J., *Technique for assessing external design of software* **21**, 2, 211 (1982); *G321-5167.*

Peck, T. G., *Worldwide systems engineering* **24**, 3/4, 182 (1985); *G321-5247.*

Penafiel, H. R., see McGee, W. C., **7**, 3/4, 342 (1968).

Pennacchi, R., see Ambrosetti, R., **23**, 4, 336 (1984).

Pennebaker, W. B., see Anastassiou, D., **22**, 1/2, 97 (1983).

Pennington, K. S., see Anastassiou, D., **22**, 1/2, 97 (1983).

Percival, L. C., and S. K. Johnson, *Network management software usability test design and implementation* **25**, 1, 92 (1986); *G321-5264.*

Pessin, F. H., see Bethke, F. J., **20**, 3, 306 (1981).

Peterson, T. I., see Parmelee, R. P., **11**, 2, 99 (1972).

Petit, A. C. R., see Boyd, D. F., **3**, 1, 51 (1964).

Phillips, G. H., see Simpson, R. O., **17**, 3, 221 (1978).

Phillips, R. J., see Attanasio, C. R., **15**, 1, 102 (1976).

Phillips, R. W., see Radice, R. A., **24**, 2, 91 (1985).

Phillips, R. W., see Mays, R. G., **24**, 2, 134 (1985).

Piccolo, F., see Angeleri, P., **27**, 4, 541 (1988).

Pierce, J. F., *A multi-item economic lot-sizing problem* **7**, 1, 47 (1968).

Pitkowsky, S. H., see Conti, C. J., **7**, 1, 2 (1968).

Plachy, E. C., see Toomey, L. J., **27**, 4, 416 (1988).

Platte, B. G., see deBry, R. K., **27**, 2, 219 (1988).

Platte, B. G., see deBry, R. K., **27**, 2, 234 (1988).

Poland, C. B., see Baldwin, F. R., **1**, 1, 64 (1962).

Pomeroy, J. W., *A guide to programming tools and techniques* **11**, 3, 234 (1972).

Pope, W. G., see Doherty, W. J., **25**, 3/4, 306 (1986).

Power, L. R., see Belady, L. A., **19**, 4, 542 (1980).

Power, L. R., see Archibald, J. L., **22**, 3, 170 (1983).

Power, L. R., *Design and use of a program execution analyzer* **22**, 3, 271 (1983); *G321-5196.*

Powning, L. F., see Halsey, J. R., **18**, 2, 223 (1979).

Pozefsky, M., see Gray, J. P., **22**, 4, 298 (1983).

Prager, J. M., *The Project Automated Librarian* **22**, 3, 214 (1983); *G321-5192.*

Prairie, P. L., and A. H. Weis, *IBM's directions in technical computing* **27**, 4, 393 (1988); *G321-5334.*

Price, J. F., see Stacy, E. W., **8**, 3, 220 (1969).

Price, T. G., see Blasgen, M. W., **20**, 1, 41 (1981).

Provetero, J., see Hippert, R. O., Jr., **14**, 1, 81 (1975).

Putzolu, G. R., see Blasgen, M. W., **20**, 1, 41 (1981).

Quarles, D. R., *Trajectory control programs in support of Apollo missions* **9**, 1, 12 (1970).

Quinnan, R. E., *The management of software engineering Part V: Software engineering management practices* **19**, 4, 466 (1980); *G321-5133.*

Racke, W. F., see Schütt, T. E., **26**, 3, 235 (1987).

Radice, R. A., N. K. Roth, A. C. O'Hara, Jr., and W. A. Ciarfella, *A programming process architecture* **24**, 2, 79 (1985); *G321-5240.*

Radice, R. A., J. T. Harding, P. E. Munnis, and R. W. Phillips, *A programming process study* **24**, 2, 91 (1985); *G321-5241.*

Radke, C. E., *Merge-sort analysis by matrix techniques* **5**, 4, 226 (1966).

Raimondi, D. L., H. M. Gladney, G. Hochweller, R. W. Martin, and L. L. Spencer, *LABS/7–a distributed real-time operating system* **15**, 1, 81 (1976); *G321-5028.*

Rain, D. W., see Gibson, D. H., **25**, 1, 36 (1986).

Rake, S. T., see Elson, M., **9**, 3, 166 (1970).

Ratica, G. B., see Voelker, J. A., **25**, 2, 181 (1986).

Rauf, R. C., see Dunfee, W. P., **27**, 3, 325 (1988).

Raver, N., and G. U. Hubbard, *Automated logical data base design: Concepts and applications* **16**, 3, 287 (1977); *G321-5055.*

Ravin, J., and M. Schatzoff, *An interactive graphics system for analysis of business decisions* **12**, 3, 238 (1973).

Rawson, F. L., III, see Kogan, M. S., **27**, 2, 90 (1988).

Rawson, F. L., III, see Cook, R. L., **27**, 2, 134 (1988).

Rechtschaffen, R. N., *Queuing simulation using a random number generator* **11**, 3, 255 (1972).

Reinsch, R., *Distributed database for SAA* **27**, 3, 362 (1988); *G321-5331.*

Reiser, M., see Anderson, H. A., Jr., **14**, 3, 246 (1975).

Reiser, M., *Interactive modeling of computer systems* **15**, 4, 309 (1976); *G321-5039.*

Rettberg, J. W., see Halpern, P., **20**, 1, 86 (1981).

Rettus, R. C., and R. A. Smith, *Accounting control of data processing* **11**, 1, 74 (1972).

Revell, R., see Nguyen, H. C., **19**, 1, 81 (1980).

Richardson, G. L., see Hodil, E. D., **25**, 2, 147 (1986).

Ricour, D. H., and V. Mei, *Internal data management techniques for DOS/360* **6**, 1, 38 (1967).

Risman, L. J., see Katz, S., **19**, 4, 505 (1980).

Robbins, F. H., see Kravitz, J. K., **25**, 1, 116 (1986).

Roberts, M. L., and P. D. Griffiths, *Design considerations for IBM Personal Computer Professional FORTRAN, an optimizing compiler* **24**, 1, 49 (1985); *G321-5237.*

Roberts, S. M., see Halpern, P., **26**, 2, 201 (1987).

Rockefeller, R. L., see Lancto, D. C., **6**, 2, 103 (1967).

Rodeh, M., see Katz, S., **19**, 4, 505 (1980).

Rogers, G. R., *A simple architecture for consistent application program design* **22**, 3, 199 (1983); *G321-5191.*

Rogers, J. G., *Structured programming for virtual storage systems* **14**, 4, 385 (1975); *G321-5023.*

Rojas, J., see Tavera, M. L., **24**, 1, 61 (1985).

Rose, D. B., and J. E. Munn, *SNA network management directions* **27**, 1, 3 (1988); *G321-5307.*

Roth, N. K., see Radice, R. A., **24**, 2, 79 (1985).

Ruggiero, J. F., and D. A. Coryell, *An auxiliary processing system for array calculations* **8**, 2, 118 (1969).

Rully, A. D., *Interactive Graphics in Data Processing A subroutine package for FORTRAN* **7**, 3/4, 248 (1968).

Rupp, B. A., see Judisch, J. M., **20**, 2, 172 (1981).

Ryan, R. R., and H. Spiller, *The C programming language and a C compiler* **24**, 1, 37 (1985); *G321-5236.*

Ryder, K. D., *A heuristic approach to task dispatching* **9**, 3, 189 (1970).

Ryder, K. D., *Optimizing program placement in virtual systems* **13**, 4, 292 (1974).

Ryder, K. D., see Albrecht, H. R., **15**, 1, 53 (1976).

Ryder, K. D., *An experimental address space isolation technique for SNA networks* **22**, 4, 367 (1983); *G321-5200.*

Saarinen, J., see Clementi, E., **27**, 4, 475 (1988).

Sahulka, R. J., see Toomey, L. J., **27**, 4, 416 (1988).

Sakaki, M., H. Samukawa, and N. Honjou, *Effective utilization of IBM 3090 large virtual storage in the numerically intensive computations of* ab initio *molecular orbitals* **27**, 4, 528 (1988); *G321-5341.*

Sakamoto, J. G., and F. W. Ball, *Supporting Business Systems Planning studies with the DB/DC Data Dictionary* **21**, 1, 54 (1982); *G321-5161.*

Sales, R., *COBOL/2: The next generation in applications programming* **27**, 2, 158 (1988); *G321-5316.*

Salisbury, C. A., see Holley, L. H., **18**, 1, 47 (1979).

Samukawa, H., *Programming style on the IBM 3090 Vector Facility considering both performance and flexibility* **27**, 4, 453 (1988); *G321-5338.*

Samukawa, H., see Sakaki, M., **27**, 4, 528 (1988).

Sandberg, G., *A primer on relational data base concepts* **20**, 1, 23 (1981); *G321-5139.*

Sarma, K. S., *An input-output econometric model* **16**, 4, 398 (1977); *G321-5060.*

Sauer, C. H., see Loucks, L. K., **26**, 4, 326 (1987).

Saul, D. N., see Holley, L. H., **18**, 1, 47 (1979).

Savir, D., and G. J. Laurer, *The characteristics and decodability of the Universal Product Code symbol* **14**, 1, 16 (1975); *G321-5002.*

Savir, D., see Metz, W. C., Jr., **14**, 1, 46 (1975).

Scarborough, R. G., see Toomey, L. J., **27**, 4, 416 (1988).

Schafer, D. J., see Mudie, M. W., **24**, 3/4, 307 (1985).

Schardt, R. M., *An MVS tuning approach* **19**, 1, 102 (1980); *G321-5118.*

Scharfenberg, K. F., P. H. Smith, Jr., and R. D. Villani, *A concordance generator* **3**, 1, 104 (1964).

Schatzoff, M., see Margolin, B. H., **10**, 4, 283 (1971).

Schatzoff, M., see Ravin, J., **12**, 3, 238 (1973).

Scherr, A. L., *Functional structure of IBM virtual storage operating systems Part II: OS/VS2-2 concepts and philosophies* **12**, 4, 382 (1973).

Scherr, A. L., *Distributed data processing* **17**, 4, 324 (1978); *G321-5078.*

Scherr, A. L., *A perspective on communications and computing* **22**, 1/2, 5 (1983); *G321-5179.*

Scherr, A. L., *Structures for networks of systems* **26**, 1, 4 (1987); *G321-5284.*

Scherr, A. L., *SAA distributed processing* **27**, 3, 370 (1988); *G321-5332.*

Schick, T., and R. F. Brockish, *The Document Interchange Architecture: A member of a family of architectures in the SNA environment* **21**, 2, 220 (1982); *G321-5168.*

Schiller, D. C., *System capacity and performance evaluation* **19**, 1, 46 (1980); *G321-5115.*

Schkolnick, M., see Blasgen, M. W., **20**, 1, 41 (1981).

Schmidt, S., see McComb, J., **27**, 4, 404 (1988).

Schmitt, E. J., see Harrison, E. S., **26**, 3, 293 (1987).

Schneider, T., see Herscovitch, H., **4**, 3, 174 (1965).

Schreiber, D. E., see Fromm, J. E., **11**, 1, 41 (1972).

Schütt, T. E., J. B. Staton III, and W. F. Racke, *Message-handling systems based on the CCITT X.400 recommendations* **26**, 3, 235 (1987); *G321-5296.*

Schultz, G. D., see Sundstrom, R. J., **26**, 1, 13 (1987).

Schultz, H., see Hanssmann, F., **10**, 4, 305 (1971).

Scopinich, C. J., see Housel, B. C., **22**, 4, 319 (1983).

Seaman, P. H., R. A. Lind, and T. L. Wilson, *On teleprocessing system design Part IV: An analysis of auxiliary storage activity* **5**, 3, 158 (1966).

Seaman, P. H., *On teleprocessing system design Part VI: The role of digital simulation* **5**, 3, 175 (1966).

Seaman, P. H., and R. C. Soucy, *Simulating operating systems* **8**, 4, 264 (1969).

Seaman, P. H., *Modeling considerations for predicting performance of CICS/VS systems* **19**, 1, 68 (1980); *G321-5116.*

Seaman, P. H., see Acker, R. D., **21**, 4, 471 (1982).

Seawright, L. H., see Meyer, R. A., **9**, 3, 199 (1970).

Seawright, L. H., and R. A. MacKinnon, *VM/370–a study of multiplicity and usefulness* **18**, 1, 4 (1979); *G321-5084.*

Selinger, P. G., see Blasgen, M. W., **20**, 1, 41 (1981).

Selinger, P. G., *Database technology* **26**, 1, 96 (1987); *G321-5288.*

Senko, M. E., E. B. Altman, M. M. Astrahan, and P. L. Fehder, *Data structures and accessing in data-base systems Part I: Evolution of information systems* **12**, 1, 30 (1973).

Senko, M. E., E. B. Altman, M. M. Astrahan, and P. L. Fehder, *Data structures and accessing in data-base systems Part II: Information organization* **12**, 1, 45 (1973).

Senko, M. E., E. B. Altman, M. M. Astrahan, and P. L. Fehder, *Data structures and accessing in data-base systems Part III: Data representations and the data independent accessing model* **12**, 1, 64 (1973).

Senko, M. E., *Data structures and data accessing in data base systems past, present, future* **16**, 3, 208 (1977); *G321-5053.*

Seward, F. K., *An application-oriented multiprocessing system Part VI: Programs for the intended application* **6**, 2, 124 (1967).

Sguazzero, P., see Kamel, A., **27**, 4, 510 (1988).

Shah, M. J., *Automatic programming for energy management using sensor based computers* **18**, 3, 457 (1979); *G321-5104.*

Shannon, A. W., see Toomey, L. J., **27**, 4, 416 (1988).

Shapiro, S., *A technique to control waiting time in a queue* **4**, 1, 53 (1965).

Shapiro, S., *Fabrication and assembly operations Part VII: Adaptive control in production planning* **4**, 3, 250 (1965).

Shaw, J. F., see Toomey, L. J., **27**, 4, 416 (1988).

Shedler, G. S., and M. M. Lehman, *Evaluation of redundancy in a parallel algorithm* **6**, 3, 142 (1967).

Shedler, G. S., and S. C. Yang, *Simulation of a model of paging system performance* **10**, 2, 113 (1971).

Shepherd, B. J., *Experimental page makeup of text with graphics on a raster printer* **19**, 3, 345 (1980); *G321-5129.*

Shibamiya, A., see Cheng, J. M., **23**, 2, 189 (1984).

Shiobara, M. O., see Sy, K. K., **26**, 2, 157 (1987).

Shipp, K. O., see Dunfee, W. P., **27**, 3, 325 (1988).

Shu, N. C., see Lum, V. Y., **21**, 3, 327 (1982).

Shukuya, S., see Sy, K. K., **26**, 2, 157 (1987).

Silberberg, M. Y., see Brennan, R. D., **6**, 4, 242 (1967).

Simons, R. M., see Neal, A. S., **23**, 1, 82 (1984).

Simpson, R. O., and C. H. Phillips, *Network job entry facility for JES2* **17**, 3, 221 (1978); *G321-5072.*

Simpson, R. O., and P. D. Hester, *The IBM RT PC ROMP processor and memory management unit architecture* **26**, 4, 346 (1987); *G321-5301.*

Singh, Y., G. M. King, and J. W. Anderson, *IBM 3090 performance: A balanced system approach* **25**, 1, 20 (1986); *G321-5259.*

Siwiec, J. E., *A high-performance DB/DC system* **16**, 2, 169 (1977); *G321-5051.*

Skatrud, R. O., see Hippert, R. O., Jr., **14**, 1, 81 (1975).

Skinner, C. E., and J. R. Asher, *Effects of storage contention on system performance* **8**, 4, 319 (1969).

Skwish, W. J., see Nguyen, H. C., **19**, 1, 81 (1980).

Slate, L., and K. Spielberg, *The Extended Control Language of MPSX/370 and possible applications* **17**, 1, 64 (1978); *G321-5064.*

Slutz, D. R., see Mattson, R. L., **9**, 2, 78 (1970).

Slutz, D. R., see Blasgen, M. W., **20**, 1, 41 (1981).

Slutz, D. R., see Chamberlin, D. D., **21**, 3, 250 (1982).

Smedley, B. S., *The development of software systems to aid in physical planning* **17**, 4, 359 (1978); *G321-5080.*

Smith, C. P., see Christensen, K., **20**, 4, 372 (1981).

Smith, E. C., Jr., *Simulation in systems engineering* **1**, 1, 33 (1962).

Smith, E. C., Jr., *A directly coupled multiprocessing system* **2**, 3/4, 218 (1963).

Smith, J. D., see Bender, G., **6**, 1, 2 (1967).

Smith, P. H., Jr., see Scharfenberg, K. F., **3**, 1, 104 (1964).

Smith, R. A., see Rettus, R. C., **11**, 1, 74 (1972).

Smith-Kerker, P. L., see Waters, F. C. H., **26**, 4, 383 (1987).

So, K., and V. Zecca, *Program locality of vectorized applications running on the IBM 3090 with Vector Facility* **27**, 4, 436 (1988); *G321-5337.*

Somerville, P. J., *Use of images in commercial and office systems* **23**, 3, 281 (1984); *G321-5225.*

Sordi, J. J., *The Query Management Facility* **23**, 2, 126 (1984); *G321-5214.*

Soucy, R. C., see Seaman, P. H., **8**, 4, 264 (1969).

Soucy, R. C., see Bailey, R. M., **22**, 4, 404 (1983).

Spencer, L. L., see Raimondi, D. L., **15**, 1, 81 (1976).

Spielberg, K., *Computation of e(x) with the use of large tables* **5**, 2, 102 (1966).

Spielberg, K., see Slate, L., **17**, 1, 64 (1978).

Spiller, H., see Ryan, R. R., **24**, 1, 37 (1985).

Spillman, T. C., see Kozuh, F. T., **23**, 3, 245 (1984).

Spivak, G. A., see Tetzlaff, W. H., **26**, 2, 215 (1987).

Stacy, E. W., L. E. Hunsinger, and J. F. Price, *Determining economic sampling plans* **8**, 3, 220 (1969).

Stanley, W. I., and H. F. Hertel, *Statistics gathering and simulation for the Apollo real-time operating system* **7**, 2, 85 (1968).

Stanley, W. I., *Measurement of system operational statistics* **8**, 4, 299 (1969).

Starbird, R. P., see Burns, N. A., **25**, 2, 236 (1986).

Staton, J. B., III, see Sundstrom, R. J., **26**, 1, 13 (1987).

Staton, J. B., III, see Schütt, T. E., **26**, 3, 235 (1987).

Stay, J. F., *HIPO and integrated program design* **15**, 2, 143 (1976); *G321-5031.*

Sternad, N., *Programming considerations for the 7750* **2**, 1, 57 (1963).

Stevens, W. P., G. J. Myers, and L. L. Constantine, *Structured design* **13**, 2, 115 (1974).

Stevens, W. P., *How data flow can improve application development productivity* **21**, 2, 162 (1982); *G321-5165.*

Stevens, W. P., see Buchwald, L. A., **27**, 3, 315 (1988).

Stevens, W. Y., *The structure of SYSTEM/360 Part II: System implementations* **3**, 2, 136 (1964).

Stevenson, D. A., see McManigal, D. F., **19**, 3, 331 (1980).

Stewart, H. M., *Performance analysis of complex communications systems* **18**, 3, 356 (1979); *G321-5100.*

Stone, R. L., see Gladney, H. M., **14**, 4, 314 (1975).

Streeter, D. N., *Cost-benefit evaluation of scientific computing services* **11**, 3, 219 (1972).

Streeter, D. N., *Centralization or dispersion of computing facilities* **12**, 3, 283 (1973).

Streeter, D. N., *Productivity of computer-dependent workers* **14**, 3, 292 (1975); *G321-5016.*

Strickland, J. P., P. P. Uhrowczik, and V. L. Watts, *IMS/VS: An evolving system* **21**, 4, 490 (1982); *G321-5178.*

Strole, N. C., see Dixon, R. C., **22**, 1/2, 47 (1983).

Strom, R. E., see Parr, F. N., **22**, 1/2, 111 (1983).

Strong, H. R., see Blasgen, M. W., **20**, 1, 41 (1981).

Suda, R., *An application-oriented multiprocessing system Part V: The diagnostic monitor* **6**, 2, 116 (1967).

Suez, J., see Branin, F. H., Jr., **4**, 1, 2 (1965).

Sullivan, T. P., *Communications Network Management enhancements for SNA networks: An overview* **22**, 1/2, 129 (1983); *G321-5187.*

Sultan, R. A., P. Kermani, G. A. Grover, T. P. Barzilai, and A. E. Baratz, *Implementing System/36 Advanced Peer-to-Peer Networking* **26**, 4, 429 (1987); *G321-5306.*

Summers, R. C., see Wood, C., **19**, 2, 229 (1980).

Summers, R. C., *An overview of computer security* **23**, 4, 309 (1984); *G321-5227.*

Sundstrom, R. J., J. B. Staton III, G. D. Schultz, M. L. Hess, G. A. Deaton, Jr., L. J. Cole, and R. M. Amy, *SNA: Current requirements and direction* **26**, 1, 13 (1987); *G321-5285.*

Sussenguth, E. H., see Falkoff, A. D., **3**, 2, 198 (1964).

Sweeney, D. W., *An analysis of floating-point addition* **4**, 1, 31 (1965).

Sy, K. K., M. O. Shiobara, M. Yamaguchi, Y. Kobayashi, S. Shukuya, and T. Tomatsu, *OSI-SNA interconnections* **26**, 2, 157 (1987); *G321-5292.*

Sykes, S. C. G., *The Realtime Interface Co-Processor Multiport/2 adapter* **27**, 2, 198 (1988); *G321-5319.*

Symonds, A. J., *Interactive Graphics in Data Processing Auxiliary-storage associative data structure for PL/I* **7**, 3/4, 229 (1968).

Symonds, A. J., *Introduction to IBM's knowledge-systems products* **25**, 2, 134 (1986); *G321-5267.*

Szczygielski, T. J., see Baily, J. H., **13**, 3, 253 (1974).

Talmadge, R. B., *Design of an integrated programming and operating system Part II: The assembly program and its language* **2**, 2, 162 (1963).

Tang, D. T., and R. T. Chien, *Coding for error control* **8**, 1, 48 (1969).

Tavera, M. L., see Alfonseca, M., **16**, 1, 18 (1977).

Tavera, M. L., M. Alfonseca, and J. Rojas, *An APL system for the IBM Personal Computer* **24**, 1, 61 (1985); *G321-5238.*

Teng, J. Z., and R. A. Gumaer, *Managing IBM Database 2 buffers to maximize performance* **23**, 2, 211 (1984); *G321-5219.*

Tennison, R. D., see Belady, L. A., **10**, 2, 143 (1971).

Tetzlaff, W. H., *State sampling of interactive VM/370 users* **18**, 1, 164 (1979); *G321-5091.*

Tetzlaff, W. H., see Bozman, G., **23**, 1, 44 (1984).

Tetzlaff, W. H., and W. M. Buco, *VM/370, Attached Processor, and multiprocessor performance study* **23**, 4, 375 (1984); *G321-5231.*

Tetzlaff, W. H., T. Beretvas, W. M. Buco, J. Greenberg, D. R. Patterson, and G. A. Spivak, *A page-swapping prototype for VM/HPO* **26**, 2, 215 (1987); *G321-5295.*

Thadhani, A. J., *Interactive user productivity* **20**, 4, 407 (1981); *G321-5156.*

Thadhani, A. J., *Factors affecting programmer productivity during application development* **23**, 1, 19 (1984); *G321-5207.*

Thomas, J. C., and J. M. Carroll, *Human factors in communication* **20**, 2, 237 (1981); *G321-5148.*

Thomas, J. C., see Branscomb, L. M., **23**, 3, 224 (1984).

Thomason, L. C., see Albrecht, H. R., **18**, 4, 526 (1979).

Thombs, H. W., see Gagliano, F. W., **7**, 3/4, 281 (1968).

Thompson, C. H., see Callaway, P. H., **11**, 3, 200 (1972).

Thompson, S., *VGA–Design choices for a new video subsystem* **27**, 2, 185 (1988); *G321-5318.*

Thorn, C. A., see Brown, D. T., **11**, 3, 186 (1972).

Tillman, C. C., see Parmelee, R. P., **11**, 2, 99 (1972).

Todd, S. J. P., *The Peterlee Relational Test Vehicle–a system overview* **15**, 4, 285 (1976); *G321-5038.*

Todd, S. J. P., see Chamberlin, D. D., **21**, 3, 250 (1982).

Tomatsu, T., see Sy, K. K., **26**, 2, 157 (1987).

Tondo, C. L., see Korn, P. A., **24**, 1, 26 (1985).

Toomey, L. J., E. C. Plachy, R. G. Scarborough, R. J. Sahulka, J. F. Shaw, and A. W. Shannon, *IBM Parallel FORTRAN* **27**, 4, 416 (1988); *G321-5336.*

Traiger, I. L., see Mattson, R. L., **9**, 2, 78 (1970).

Traiger, I. L., see Blasgen, M. W., **20**, 1, 41 (1981).

Traub, W. G., see Engel, G. H., **18**, 3, 402 (1979).

Trimble, J., see White, B., **3**, 1, 79 (1964).

Tuchman, W. L., see Ehrsam, W. F., **17**, 2, 106 (1978).

Tucker, S. G., *Microprogram control for SYSTEM/360* **6**, 4, 222 (1967).

Tucker, S. G., *The IBM 3090 system: An overview* **25**, 1, 4 (1986); *G321-5258.*

Tuckerman, B., see Konheim, A. G., **19**, 2, 253 (1980).

Tunis, C. J., see Griffin, J. S., Jr., **2**, 3/4, 248 (1963).

Tunkel, J. A., see Cook, R. L., **27**, 2, 134 (1988).

Turnburke, V. P., Jr., *Sequential data processing design* **2**, 1, 37 (1963).

Uhlir, S., *Enabling the user interface* **27**, 3, 306 (1988); *G321-5327.*

Uhrowczik, P. P., *Data Dictionary/Directories* **12**, 4, 332 (1973).

Uhrowczik, P. P., see Strickland, J. P., **21**, 4, 490 (1982).

Van Leer, P., *Top-down development using a program design language* **15**, 2, 155 (1976); *G321-5032.*

Van Slyke, D., see Mishelevich, D. J., **19**, 4, 478 (1980).

Van Woerkom, H. M., see Milliken, K. R., **25**, 2, 159 (1986).

Veber, R. A. K., see Wade, R. D., **8**, 1, 28 (1969).

Velasco, C. R., *A general purpose digital simulator and examples of its application Part II: Simulation of a telephone intercept system* **3**, 1, 35 (1964).

Villani, R. D., see Scharfenberg, K. F., **3**, 1, 104 (1964).

Voelker, J. A., and G. B. Ratica, *The genesis of a knowledge-based expert system* **25**, 2, 181 (1986); *G321-5270.*

Vogelsberg, R. E., see Jephson, J. S., **8**, 3, 178 (1969).

Wade, B. W., see Blasgen, M. W., **20**, 1, 41 (1981).

Wade, B. W., see Chamberlin, D. D., **21**, 3, 250 (1982).

Wade, R. D., G. P. Cawsey, and R. A. K. Veber, *A teleprocessing approach using standard equipment* **8**, 1, 28 (1969).

Wagner, R. E., *Indexing design considerations* **12**, 4, 351 (1973).

Wahi, P. N., *A general management business simulation in APL* **11**, 2, 169 (1972).

Waite, N. B., see Milliken, K. R., **25**, 2, 159 (1986).

Waldbaum, G., see Friedman, H. P., **14**, 4, 340 (1975).

Waldbaum, G., see Konheim, A. G., **19**, 2, 253 (1980).

Waldbaum, G., see Chess, D. M., **20**, 4, 424 (1981).

Waldbaum, G., see Koved, L., **25**, 1, 105 (1986).

Wall, W. A., see Duke, K. A., **24**, 1, 14 (1985).

Walsh, H. F., see Gibson, D. H., **25**, 1, 36 (1986).

Walston, C. E., and C. P. Felix, *A method of programming measurement and estimation* **16**, 1, 54 (1977); *G321-5045.*

Waters, F. C. H., *Design of the IBM 8100 Data Base and Transaction Management System–DTMS* **18**, 4, 565 (1979); *G321-5110.*

Waters, F. C. H., R. G. Bias, and P. L. Smith-Kerker, *AIX usability enhancements and human factors* **26**, 4, 383 (1987); *G321-5303.*

Watson, H. K., see Anthony, A. L., **11**, 4, 316 (1972).

Watson, S. L., *OS/2 Query Manager overview and prompted interface* **27**, 2, 119 (1988); *G321-5314.*

Watts, V. L., see Strickland, J. P., **21**, 4, 490 (1982).

Weber, J. C., see Hagamen, W. D., **11**, 4, 278 (1972).

Weber, J. C., see Hagamen, W. D., **14**, 2, 102 (1975).

Weingarten, R. A., *An integrated approach to centralized communications network management* **18**, 4, 484 (1979); *G321-5106.*

Weingarten, R. A., see Benjamin, J. H., **22**, 4, 344 (1983).

Weingarten, R. A., and E. E. Iacobucci, *Logical problem determination for SNA networks* **22**, 4, 387 (1983); *G321-5201.*

Weingarten, R. A., see Jaffe, J. M., **22**, 4, 417 (1983).

Weis, A. H., see Prairie, P. L., **27**, 4, 393 (1988).

Welch, P. D., *On the reliability of polymorphic systems* **4**, 1, 43 (1965).

Weller, D. L., E. D. Carlson, G. M. Giddings, F. P. Palermo, R. Williams, and S. N. Zilles, *Software architecture for graphical interaction* **19**, 3, 314 (1980); *G321-5127.*

Wheeler, E. F., and A. G. Ganek, *Introduction to Systems Application Architecture* **27**, 3, 250 (1988); *G321-5323.*

Wheeler, T. F., Jr., *OS/VS1 concepts and philosophies* **13**, 3, 213 (1974).

Wheeler, W. R., see Benjamin, J. H., **22**, 4, 344 (1983).

White, B., and J. Trimble, *Design of an integrated programming and operating system Part VI: Implementation on the 7040/44 data processing system* **3**, 1, 79 (1964).

White, W. W., and A. M. Bomberault, *A network algorithm for empty freight car allocation* **8**, 2, 147 (1969).

Williams, K. C., see Esau, L. R., **5**, 3, 142 (1966).

Williams, R., see Weller, D. L., **19**, 3, 314 (1980).

Williams, R. J., see Margopoulos, W. P., **5**, 3, 134 (1966).

Williams, R. L., see Cook, R. L., **27**, 2, 134 (1988).

Wilson, T. L., see Seaman, P. H., **5**, 3, 158 (1966).

Wilson, T. L., see Clark, R. S., **25**, 1, 63 (1986).

Wilson, W. G., *Prolog for applications programming* **25**, 2, 190 (1986); *G321-5271.*

Wimbrow, J. H., *A large-scale interactive administrative system* **10**, 4, 260 (1971).

Witt, B. I., *Dynamic storage allocation for a real-time system* **2**, 3/4, 230 (1963).

Witt, B. I., *The functional structure of OS/360 Part II: Job and task management* **5**, 1, 12 (1966).

Wladawsky-Berger, I., see Goldstein, B. C., **23**, 3, 236 (1984).

Wolfe, R. F., see Cordell, R. Q., II, **26**, 4, 361 (1987).

Wolford, D. E., *Application enabling in SAA* **27**, 3, 301 (1988); *G321-5326.*

Wong, D. J., see Chang, W., **4**, 2, 162 (1965).

Wood, C., E. B. Fernandez, and R. C. Summers, *Data base security: requirements, policies, and models* **19**, 2, 229 (1980); *G321-5124.*

Wood, C. E., see Green, P. E., Jr., **26**, 4, 414 (1987).

Woodrum, L. J., *Internal sorting with minimal comparing* **8**, 3, 189 (1969).

Woodrum, L. J., *A model of floating buffering* **9**, 2, 118 (1970).

Worley, E. L., see Gladney, H. M., **14**, 3, 212 (1975).

Worthington, P. S., see Cheng, J. M., **23**, 2, 189 (1984).

Yagil, S., *Generation of input data for simulations* **2**, 3/4, 288 (1963).

Yamada, H., see Galli, E. J., **6**, 3, 192 (1967).

Yamaguchi, M., see Sy, K. K., **26**, 2, 157 (1987).

Yang, S. C., see Shedler, G. S., **10**, 2, 113 (1971).

Yardeni, L. A., *Algorithms for traffic-signal control* **4**, 2, 148 (1965).

Yelavich, B. M., *Customer Information Control System–An evolving system facility* **24**, 3/4, 264 (1985); *G321-5253.*

Yost, R. A., see Blasgen, M. W., **20**, 1, 41 (1981).

Young, G. E., see George, F. D., **21**, 2, 179 (1982).

Yuval, A., *System contention analysis–An alternate approach to system tuning* **19**, 2, 208 (1980); *G321-5123.*

Zachman, J. A., *Business Systems Planning and Business Information Control Study: A comparison* **21**, 1, 31 (1982); *G321-5160.*

Zachman, J. A., *A framework for information systems architecture* **26**, 3, 276 (1987); *G321-5298.*

Zecca, V., see So, K., **27**, 4, 436 (1988).

Ziegler, K., Jr., *A distributed information system study* **18**, 3, 374 (1979); *G321-5101.*

Zilles, S. N., see Weller, D. L., **19**, 3, 314 (1980).

Zimmerman, L. L., see Day, C. W., **7**, 3/4, 373 (1968).

Zimowski, M. R., see Hammond, K. R., **23**, 2, 151 (1984).

Zloof, M. M., *Query-by-Example: A data base language* **16**, 4, 324 (1977); *G321-5056.*

Zloof, M. M., *Office-by-Example: A business language that unifies data and word processing and electronic mail* **21**, 3, 272 (1982); *G321-5170.*

IBM Systems Journal Subject Index 1972–1988

Architecture

Channel and direct access device architecture, *Brown*, **11**, 3, 186 (1972)

Functional structure of IBM virtual storage operating systems Part III: Architecture and design of DOS/VS, *Birch*, **12**, 4, 401 (1973)

Overview of the Supermarket System and the Retail Store System, *McEnroe*, **14**, 1, 3 (1975)

Systems Network Architecture: An overview, *McFadyen*, **15**, 1, 4 (1976)

The transmission subsystem in Systems Network Architecture, *Cullum*, **15**, 1, 24 (1976)

Service levels: A concept for the user and the computer center, *Lewis*, **15**, 4, 328 (1976)

The information management system IMS/VS Part I: General structure and operation, *McGee*, **16**, 2, 84 (1977)

IBM's Santa Teresa Laboratory–Architectural design for program development, *McCue*, **17**, 1, 4 (1978)

Administrative control of computing service, *Gladney*, **17**, 2, 151 (1978)

An introduction to network architectures and protocols, *Green*, **18**, 2, 202 (1979)

Software architecture for graphical interaction, *Weller*, **19**, 3, 314 (1980)

Architecture of the IBM 3277 Graphics Attachment, *McManigal*, **19**, 3, 331 (1980)

The management of software engineering Part II: Software engineering program, *O'Neill*, **19**, 4, 421 (1980)

Application development system: The software architecture of the IBM Health Care Support/DL/I-Patient Care System, *Mishelevich*, **19**, 4, 478 (1980)

Electronic information interchange in an office environment, *DeSousa*, **20**, 1, 4 (1981)

System R: An architectural overview, *Blasgen*, **20**, 1, 41 (1981)

SNA flow control: Architecture and implementation, *George*, **21**, 2, 179 (1982)

The Document Interchange Architecture: A member of a family of architectures in the SNA environment, *Schick*, **21**, 2, 220 (1982)

A simple architecture for consistent application program design, *Rogers*, **22**, 3, 199 (1983)

Architecture prototyping in the software engineering environment, *Beregi*, **23**, 1, 4 (1984)

Ease of use: A system design challenge, *Branscomb*, **23**, 3, 224 (1984)

System/370 capability in a desktop computer, *Kozuh*, **23**, 3, 245 (1984)

Architecture implications in the design of microprocessors, *Matick*, **23**, 3, 264 (1984)

An application analyzer, *Ambrosetti*, **23**, 4, 336 (1984)

A programming process architecture, *Radice*, **24**, 2, 79 (1985)

Information System Model and Architecture Generator, *Hein*, **24**, 3/4, 213 (1985)

Customer Information Control System–An evolving system facility, *Yelavich*, **24**, 3/4, 264 (1985)

The System Planning Grid: A model for building integrated information systems, *Buckelew*, **24**, 3/4, 294 (1985)

An information technology architecture for change, *Mudie*, **24**, 3/4, 307 (1985)

The IBM 3090 system: An overview, *Tucker*, **25**, 1, 4 (1986)

The IBM System/370 vector architecture, *Buchholz*, **25**, 1, 51 (1986)

Systems architecture in transition–An overview, *Lorin*, **25**, 3/4, 256 (1986)

Impact of memory systems on computer architecture and system organization, *Matick*, **25**, 3/4, 274 (1986)

IBM small-system architecture and design–Past, present, and future, *Henry*, **25**, 3/4, 321 (1986)

A perspective on the 801/Reduced Instruction Set Computer, *Hopkins*, **26**, 1, 107 (1987)

A framework for information systems architecture, *Zachman*, **26**, 3, 276 (1987)

The IBM RT PC ROMP processor and memory management unit architecture, *Simpson*, **26**, 4, 346 (1987)

An architecture for a business and information system, *Devlin*, **27** 1, 60 (1988)

COBOL/2: The next generation in applications programming, *Sales*, **27**, 2, 158 (1988)

Understanding device drivers in Operating System/2, *Mizell*, **27**, 2, 170 (1988)

Introduction to Systems Application Architecture, *Wheeler*, **27**, 3, 250 (1988)

Common Communications Support in Systems Application Architecture, *Ahuja*, **27**, 3, 264 (1988)

Common User Access–A consistent and usable human-computer interface for the SAA environments, *Berry*, **27**, 3, 281 (1988)

Application enabling in SAA, *Wolford*, **27**, 3, 301 (1988)

Enabling the user interface, *Uhlir*, **27**, 3, 306 (1988)

Integrating applications with SAA, *Buchwald*, **27**, 3, 315 (1988)

Designing SAA applications and user interfaces, *Dunfee*, **27**, 3, 325 (1988)

Distributed files for SAA, *Demers*, **27**, 3, 348 (1988)

Distributed database for SAA, *Reinsch*, **27**, 3, 362 (1988)

SAA distributed processing, *Scherr*, **27**, 3, 370 (1988)

The Cross System Product application generator: An evolution, *Haynes*, **27**, 3, 384 (1988)

Business Planning

Concepts of financial models, *Kingston*, **12**, 2, 113 (1973)

A guide to financial planning tools and techniques, *Dzielinski*, **12**, 2, 126 (1973)

Planning-data systems, *Lande*, **12**, 2, 145 (1973)

Financial modeling on small systems, *Gordon*, **12**, 2, 161 (1973)

Interactive simulation for banking, *Brown*, **12**, 2, 172 (1973)

Forecasting techniques, *Aiso*, **12**, 2, 187 (1973)

An interactive graphics system for analysis of business decisions, *Ravin*, **12**, 3, 238 (1973)

Overview of the capacity planning process for production data processing, *Bronner*, **19**, 1, 4 (1980)

A capacity planning methodology, *Cooper*, **19**, 1, 28 (1980)

System capacity and performance evaluation, *Schiller*, **19**, 1, 46 (1980)

Modeling considerations for predicting performance of CICS/VS systems, *Seaman*, **19**, 1, 68 (1980)

The role of detailed simulation in capacity planning, *Nguyen*, **19**, 1, 81 (1980)

Strategies for information requirements determination, *Davis*, **21**, 1, 4 (1982)

Business Systems Planning and Business Information Control Study: A comparison, *Zachman*, **21**, 1, 31 (1982)

Supporting Business Systems Planning studies with the DB/DC Data Dictionary, *Sakamoto*, **21**, 1, 54 (1982)

Enterprise information analysis: Cost-benefit analysis and the data-managed system, *Parker*, **21**, 1, 108 (1982)

An application analyzer, *Ambrosetti*, **23**, 4, 336 (1984)

Information System Model and Architecture Generator, *Hein*, **24**, 3/4, 213 (1985)

An approach to high availability in high-transaction-rate systems, *Brooks*, **24**, 3/4, 279 (1985)

Communications

Synchronous data link control: A perspective, *Donnan*, **13**, 2, 140 (1974)

The role of the Network Control Program in Systems Network Architecture, *Hobgood*, **15**, 1, 39 (1976)

The information management system IMS/VS Part IV: Data communication facilities, *McGee*, **16**, 2, 136 (1977)

The information management system IMS/VS Part V: Transaction processing facilities, *McGee*, **16**, 2, 148 (1977)

A formal approach for communication between logically isolated virtual machines, *Jensen*, **18**, 1, 71 (1979)

Computing and communications–A perspective of the evolving environment, *Branscomb*, **18**, 2, 189 (1979)

Evolution of a laboratory communication network, *Moore*, **18**, 2, 315 (1979)

A perspective on communications and computing, *Scherr*, **22**, 1/2, 5 (1983)

A token-ring network for local data communications, *Dixon*, **22**, 1/2, 47 (1983)

A Satellite Communications Controller, *Fennel*, **22**, 1/2, 81 (1983)

Series/1-based videoconferencing system, *Anastassiou*, **22**, 1/2, 97 (1983)

NIL: A high-level language for distributed systems programming, *Parr*, **22**, 1/2, 111 (1983)

Workstations and mainframe computers working together, *Kravitz*, **25**, 1, 116 (1986)

Open Systems Interconnection, *Aschenbrenner*, **25**, 3/4, 369 (1986)

An advanced voice/data telephone switching system, *Kasson*, **25**, 3/4, 380 (1986)

Data communications: The implications of communication systems for protocol design, *Goldstein*, **26**, 1, 122 (1987)

OSI-SNA interconnections, *Sy*, **26**, 2, 157 (1987)

Message-handling systems based on the CCITT X.400 recommendations, *Schütt*, **26**, 3, 235 (1987)

Specification and implementation of an ISO session layer, *Fleischmann*, **26**, 3, 255 (1987)

The Realtime Interface Co-Processor Multiport/2 adapter, *Sykes*, **27**, 2, 198 (1988)

Common Communications Support in Systems Application Architecture, *Ahuja*, **27**, 3, 264 (1988)

Database

Data structures and accessing in data-base systems Part I: Evolution of information systems, *Senko*, **12**, 1, 30 (1973)

Data structures and accessing in data-base systems Part II: Information organization, *Senko*, **12**, 1, 45 (1973)

Data structures and accessing in data-base systems Part III: Data representations and the data independent accessing model, *Senko*, **12**, 1, 64 (1973)

Data Dictionary/Directories, *Uhrowczik*, **12**, 4, 332 (1973)

Performance analysis for the Skylab terminal system, *Mancini*, **13**, 2, 94 (1974)

An access control mechanism for computing resources, *Gladney*, **14**, 3, 212 (1975)

Generalized audit trail requirements and concepts for data base applications, *Bjork*, **14**, 3, 229 (1975)

The Peterlee Relational Test Vehicle–a system overview, *Todd*, **15**, 4, 285 (1976)

A user-oriented data-base retrieval system, *Jones*, **16**, 1, 4 (1977)

The information management system IMS/VS Part I: General structure and operation, *McGee*, **16**, 2, 84 (1977)

The information management system IMS/VS Part II: Data base facilities, *McGee*, **16**, 2, 96 (1977)

The information management system IMS/VS Part III: Batch processing facilities, *McGee*, **16**, 2, 123 (1977)

The information management system IMS/VS Part IV: Data communication facilities, *McGee*, **16**, 2, 136 (1977)

The information management system IMS/VS Part V: Transaction processing facilities, *McGee*, **16**, 2, 148 (1977)

A high-performance DB/DC system, *Siwiec*, **16**, 2, 169 (1977)

Data structures and data accessing in data base systems past, present, future, *Senko*, **16**, 3, 208 (1977)

Automated logical data base design: Concepts and applications, *Raver*, **16**, 3, 287 (1977)

Query-by-Example: A data base language, *Zloof*, **16**, 4, 324 (1977)

Design techniques for a user controlled DB/DC system, *Heyne*, **16**, 4, 344 (1977)

Storage and access in relational data bases, *Blasgen*, **16**, 4, 363 (1977)

Design of the IBM 8100 Data Base and Transaction Management System–DTMS, *Waters*, **18**, 4, 565 (1979)

Data base security: requirements, policies, and models, *Wood*, **19**, 2, 229 (1980)

A primer on relational data base concepts, *Sandberg*, **20**, 1, 23 (1981)

System R: An architectural overview, *Blasgen*, **20**, 1, 41 (1981)

IMS/VS: An evolving system, *Strickland*, **21**, 4, 490 (1982)

The system architecture of EAS-E: An integrated programming and data base language, *Pazel*, **22**, 3, 188 (1983)

An overview of three relational data base products, *Kahn*, **23**, 2, 100 (1984)

IBM Database 2 overview, *Haderle*, **23**, 2, 112 (1984)

The Query Management Facility, *Sordi*, **23**, 2, 126 (1984)

TSO Attach: A multipurpose communication channel to IBM Database 2, *Hammond*, **23**, 2, 151 (1984)

IBM Database 2 in an Information Management System environment, *Dash*, **23**, 2, 165 (1984)

Data recovery in IBM Database 2, *Crus*, **23**, 2, 178 (1984)

IBM Database 2 performance: Design, implementation, and tuning, *Cheng*, **23**, 2, 189 (1984)

Managing IBM Database 2 buffers to maximize performance, *Teng*, **23**, 2, 211 (1984)

HONE: The IBM marketing support system, *Boos*, **24**, 3/4, 189 (1985)

Database technology, *Selinger*, **26**, 1, 96 (1987)

A large-scale computer conferencing system, *Chess*, **26**, 1, 138 (1987)

An architecture for a business and information system, *Devlin*, **27**, 1, 60 (1988)

OS/2 EE Database Manager overview and technical highlights, *Chang*, **27**, 2, 105 (1988)

OS/2 Query Manager overview and prompted interface, *Watson*, **27**, 2, 119 (1988)

Distributed files for SAA, *Demers*, **27**, 3, 348 (1988)

Distributed database for SAA, *Reinsch*, **27**, 3, 362 (1988)

Distributed Processing

Distributed data processing, *Scherr*, **17**, 4, 324 (1978)

A distributed information system study, *Ziegler*, **18**, 3, 374 (1979)

An operating system for distributed processing–DPPX, *Kiely*, **18**, 4, 507 (1979)

I/O facilities of the Distributed Processing Programming Executive (DPPX), *Albrecht*, **18**, 4, 526 (1979)

Data Management for the Distributed Processing Programming Executive (DPPX), *Fitzgerald*, **18**, 4, 547 (1979)

Distributed processing: An assessment, *Lorin*, **18**, 4, 582 (1979)

Logical distribution of applications and data, *Baker*, **19**, 2, 171 (1980)

Distributed processing communications software support for operation within an SNA network, *Harrison*, **19**, 2, 192 (1980)

Modeling distributed processing across multiple CICS/VS sites, *Acker*, **21**, 4, 471 (1982)

SNA Distribution Services, *Housel*, **22**, 4, 319 (1983)

HONE: The IBM marketing support system, *Boos*, **24**, 3/4, 189 (1985)

Performance considerations for a distributed data processing system designed for high availability, *Agassi*, **24**, 3/4, 200 (1985)

Distributed files for SAA, *Demers*, **27**, 3, 348 (1988)

Distributed database for SAA, *Reinsch*, **27**, 3, 362 (1988)

SAA distributed processing, *Scherr*, **27**, 3, 370 (1988)

Essay

Computing and communications–A perspective of the evolving environment, *Branscomb*, **18**, 2, 189 (1979)

Potential technology implications for computers and telecommunications in the 1980s, *Frazer*, **18**, 2, 333 (1979)

Distributed processing: An assessment, *Lorin*, **18**, 4, 582 (1979)

How a computer should talk to people, *Dean*, **21**, 4, 424 (1982)

A perspective on communications and computing, *Scherr*, **22**, 1/2, 5 (1983)

Communications Network Management enhancements for SNA networks: An overview, *Sullivan*, **22**, 1/2, 129 (1983)

An overview of three relational data base products, *Kahn*, **23**, 2, 100 (1984)

Ease of use: A system design challenge, *Branscomb*, **23**, 3, 224 (1984)

General Applications

Virtual machine computing in an engineering environment, *McGrath*, **11**, 2, 131 (1972)

Numerical control for machining complex surfaces, *Almond*, **11**, 2, 150 (1972)

Uses of virtual storage systems in a scientific environment, *Callaway*, **11**, 3, 200 (1972)

Design features of a real-time check-clearing system, *Banham*, **11**, 4, 329 (1972)

A program generator, *Hagamen*, **14**, 2, 102 (1975)

The IBM 5100 and the Research Device Coupler–A personal laboratory automation system, *Cole*, **16**, 1, 41 (1977)

A high-performance DB/DC system, *Siwiec*, **16**, 2, 169 (1977)

An input-output econometric model, *Sarma*, **16**, 4, 398 (1977)

The development of software systems to aid in physical planning, *Smedley*, **17**, 4, 359 (1978)

Automatic programming for energy management using sensor based computers, *Shah*, **18**, 3, 457 (1979)

Design of the IBM 8100 Data Base and Transaction Management System–DTMS, *Waters*, **18**, 4, 565 (1979)

A graphic interactive application monitor, *Bleher*, **19**, 3, 382 (1980)

The Modular Application Customizing System, *Gordon*, **19**, 4, 521 (1980)

Engineering and scientific processing on the IBM 3090, *Gibson*, **25**, 1, 36 (1986)

The numeric representation of knowledge and logic–Two artificial intelligence applications in medical education, *Hagamen*, **25**, 2, 207 (1986)

Computer processing of dates outside the twentieth century, *Ohms*, **25**, 2, 244 (1986)

Robotics, *Korein*, **26**, 1, 55 (1987)

Writing an Operating System/2 application, *Cook*, **27**, 2, 134 (1988)

COBOL/2: The next generation in applications programming, *Sales*, **27**, 2, 158 (1988)

Introduction to Systems Application Architecture, *Wheeler*, **27**, 3, 250 (1988)

Common User Access–A consistent and usable human-computer interface for the SAA environments, *Berry*, **27**, 3, 281 (1988)

Application enabling in SAA, *Wolford*, **27**, 3, 301 (1988)

Enabling the user interface, *Uhlir*, **27**, 3, 306 (1988)

Integrating applications with SAA, *Buchwald*, **27**, 3, 315 (1988)

Designing SAA applications and user interfaces, *Dunfee*, **27**, 3, 325 (1988)

Distributed database for SAA, *Reinsch*, **27**, 3, 362 (1988)

SAA distributed processing, *Scherr*, **27**, 3, 370 (1988)

The Cross System Product application generator: An evolution, *Haynes*, **27**, 3, 384 (1988)

Seismic computations on the IBM 3090 Vector Multiprocessor, *Kamel*, **27**, 4, 510 (1988)

PAM-CRASH on the IBM 3090/VF: An integrated environment for crash analysis, *Angeleri*, **27**, 4, 541 (1988)

Interactive computations and display of characteristics of the radiation scattered by a sphere: A demonstration for PS/2 Model 80, *Halpern,* **27,** 4, 561 (1988)

Graphics and Image

System aspects of large-problem computation and display, *Fromm,* **11,** 1, 41 (1972)

An interactive graphics system for analysis of business decisions, *Ravin,* **12,** 3, 238 (1973)

The characteristics and decodability of the Universal Product Code symbol, *Savir,* **14,** 1, 16 (1975)

Experiments in computer-aided graphic expression, *Musgrave,* **17,** 3, 241 (1978)

The development of software systems to aid in physical planning, *Smedley,* **17,** 4, 359 (1978)

Interactive graphics today, *Burchi,* **19,** 3, 292 (1980)

Software architecture for graphical interaction, *Weller,* **19,** 3, 314 (1980)

Architecture of the IBM 3277 Graphics Attachment, *McManigal,* **19,** 3, 331 (1980)

Experimental page makeup of text with graphics on a raster printer, *Shepherd,* **19,** 3, 345 (1980)

A high-resolution computer graphics system, *Handelman,* **19,** 3, 356 (1980)

An APL approach to presentation graphics, *Niehoff,* **19,** 3, 367 (1980)

A graphic interactive application monitor, *Bleher,* **19,** 3, 382 (1980)

Use of images in commercial and office systems, *Somerville,* **23,** 3, 281 (1984)

Standardized graphics on the IBM Personal Computer, *Clarkson,* **24,** 1, 3 (1985)

A professional graphics controller, *Duke,* **24,** 1, 14 (1985)

The evolution of printers and displays, *Mayadas,* **25,** 3/4, 399 (1986)

Visual interpretation of complex data, *Farrell,* **26,** 2, 174 (1987)

VGA–Design choices for a new video subsystem, *Thompson,* **27,** 2, 185 (1988)

An introduction to typographic fonts and digital font resources, *Griffee,* **27,** 2, 206 (1988)

Advanced Function Printing: A tutorial, *deBry,* **27,** 2, 219 (1988)

Architectures of Advanced Function Printing, *deBry,* **27,** 2, 234 (1988)

Human Factors

User behavior on an interactive computer system, *Boies,* **13,** 1, 2 (1974)

The role of the operator in the Supermarket and Retail Store Systems, *Antonelli,* **14,** 1, 35 (1975)

Procedures of the Human Factors Center at San Jose, *Hirsch,* **20,** 2, 123 (1981)

Effects of manual style on performance in education and machine maintenance, *Judisch,* **20,** 2, 172 (1981)

Natural language programming: Styles, strategies, and contrasts, *Miller,* **20,** 2, 184 (1981)

Human factors in the development of a family of plant data communication terminals, *Ominsky,* **20,** 2, 216 (1981)

Human factors in communication, *Thomas,* **20,** 2, 237 (1981)

Software simulation as a tool for usable product design, *Clark,* **20,** 3, 272 (1981)

Improving system usability for business professionals, *Helander,* **20,** 3, 294 (1981)

Improving the usability of programming publications, *Bethke,* **20,** 3, 306 (1981)

Interactive user productivity, *Thadhani,* **20,** 4, 407 (1981)

How a computer should talk to people, *Dean,* **21,** 4, 424 (1982)

Speech filing–An office system for principals, *Gould,* **23,** 1, 65 (1984)

Playback: A method for evaluating the usability of software and its documentation, *Neal,* **23,** 1, 82 (1984)

Ease of use: A system design challenge, *Branscomb,* **23,** 3, 224 (1984)

The System Usability Process for Network Management Products, *Gottschalk,* **25,** 1, 83 (1986)

Network management software usability test design and implementation, *Percival,* **25,** 1, 92 (1986)

Tools for building advanced user interfaces, *Bennett,* **25,** 3/4, 354 (1986)

AIX usability enhancements and human factors, *Waters,* **26,** 4, 383 (1987)

Common User Access–A consistent and usable human-computer interface for the SAA environments, *Berry,* **27,** 3, 281 (1988)

Knowledge-Based Systems

Introduction to IBM's knowledge-systems products, *Symonds,* **25,** 2, 134 (1986)

Knowledge-based systems in the commercial environment, *Hodil,* **25,** 2, 147 (1986)

YES/MVS and the automation of operations for large computer complexes, *Milliken,* **25,** 2, 159 (1986)

The genesis of a knowledge-based expert system, *Voelker,* **25,** 2, 181 (1986)

Prolog for applications programming, *Wilson,* **25,** 2, 190 (1986)

The numeric representation of knowledge and logic–Two artificial intelligence applications in medical education, *Hagamen,* **25,** 2, 207 (1986)

The Portable Inference Engine: Fitting significant expertise into small systems, *Burns,* **25,** 2, 236 (1986)

Languages

Numerical control for machining complex surfaces, *Almond,* **11,** 2, 150 (1972)

Encoding verbal information as unique numbers, *Hagamen,* **11,** 4, 278 (1972)

Design of a checkout compiler, *Marks,* **12,** 3, 315 (1973)

Top-down development using a program design language, *Van Leer,* **15,** 2, 155 (1976)

Composite design facilities of six programming languages, *Myers,* **15,** 3, 212 (1976)

An APL emulator on System/370, *Hassitt,* **15,** 4, 358 (1976)

An APL interpreter and system for a small computer, *Alfonseca,* **16,** 1, 18 (1977)

Query-by-Example: A data base language, *Zloof,* **16,** 4, 324 (1977)

A method for the time analysis of programs, *de Freitas,* **17,** 1, 26 (1978)

The Extended Control Language of MPSX/370 and possible applications, *Slate,* **17,** 1, 64 (1978)

Office-by-Example: A business language that unifies data and word processing and electronic mail, *Zloof,* **21,** 3, 272 (1982)

NIL: A high-level language for distributed systems programming, *Parr,* **22,** 1/2, 111 (1983)

The system architecture of EAS-E: An integrated programming and data base language, *Pazel,* **22,** 3, 188 (1983)

The design of the REXX language, *Cowlishaw,* **23,** 4, 326 (1984)

The C programming language and a C compiler, *Ryan,* **24,** 1, 37 (1985)

Design considerations for IBM Personal Computer Professional FORTRAN, an optimizing compiler, *Roberts,* **24,** 1, 49 (1985)

An APL system for the IBM Personal Computer, *Tavera,* **24,** 1, 61 (1985)

Prolog for applications programming, *Wilson,* **25,** 2, 190 (1986)

IBM Parallel FORTRAN, *Toomey*, **27**, 4, 416 (1988)

Mathematical Methods

System aspects of large-problem computation and display, *Fromm*, **11**, 1, 41 (1972)

Numerical control for machining complex surfaces, *Almond*, **11**, 2, 150 (1972)

Encoding verbal information as unique numbers, *Hagamen*, **11**, 4, 278 (1972)

Optimizing program placement in virtual systems, *Ryder*, **13**, 4, 292 (1974)

Elements of probability for system design, *Allen*, **13**, 4, 325 (1974)

The characteristics and decodability of the Universal Product Code symbol, *Savir*, **14**, 1, 16 (1975)

Elements of queuing theory for system design, *Allen*, **14**, 2, 161 (1975)

The Extended Control Language of MPSX/370 and possible applications, *Slate*, **17**, 1, 64 (1978)

Solving the installation scheduling problem using mixed integer linear programming, *Chen*, **17**, 1, 82 (1978)

State sampling of interactive VM/370 users, *Tetzlaff*, **18**, 1, 164 (1979)

A system for constructing linear programming models, *Katz*, **19**, 4, 505 (1980)

Analytic queuing model for CICS capacity planning, *Deitch*, **21**, 4, 454 (1982)

Software reliability analysis, *Misra*, **22**, 3, 262 (1983)

Analysis of free-storage algorithms–revisited, *Bozman*, **23**, 1, 44 (1984)

IBM's directions in technical computing, *Prairie*, **27**, 4, 393 (1988)

Engineering and Scientific Subroutine Library for the IBM 3090 Vector Facility, *McComb*, **27**, 4, 404 (1988)

IBM Parallel FORTRAN, *Toomey*, **27**, 4, 416 (1988)

Programming style on the IBM 3090 Vector Facility considering both performance and flexibility, *Samukawa*, **27**, 4, 453 (1988)

ICAP 3090: Parallel processing for large-scale scientific and engineering problems, *Clementi*, **27**, 4, 475 (1988)

Seismic computations on the IBM 3090 Vector Multiprocessor, *Kamel*, **27**, 4, 510 (1988)

Effective utilization of IBM 3090 large virtual storage in the numerically intensive computations of *ab initio* molecular orbitals, *Sakaki*, **27**, 4, 528 (1988)

PAM-CRASH on the IBM 3090/VF: An integrated environment for crash analysis, *Angeleri*, **27**, 4, 541 (1988)

Interactive computations and display of characteristics of the radiation scattered by a sphere: A demonstration for PS/2 Model 80, *Halpern*, **27**, 4, 561 (1988)

Management

Chief programmer team management of production programming, *Baker*, **11**, 1, 56 (1972)

Accounting control of data processing, *Rettus*, **11**, 1, 74 (1972)

A general management business simulation in APL, *Wahi*, **11**, 2, 169 (1972)

Cost-benefit evaluation of scientific computing services, *Streeter*, **11**, 3, 219 (1972)

Net change material requirements planning, *Orlicky*, **12**, 1, 2 (1973)

Centralization or dispersion of computing facilities, *Streeter*, **12**, 3, 283 (1973)

The Power Profile–An installation management tool, *Laird*, **14**, 3, 264 (1975)

Computing center optimization by a pricing-priority policy, *Ghanem*, **14**, 3, 272 (1975)

Productivity of computer-dependent workers, *Streeter*, **14**, 3, 292 (1975)

Computer installation accounting, *Gladney*, **14**, 4, 314 (1975)

Evaluating system changes under uncontrolled workloads: a case study, *Friedman*, **14**, 4, 340 (1975)

Service levels: A concept for the user and the computer center, *Lewis*, **15**, 4, 328 (1976)

Solving the installation scheduling problem using mixed integer linear programming, *Chen*, **17**, 1, 82 (1978)

A cryptographic key management scheme for implementing the Data Encryption Standard, *Ehrsam*, **17**, 2, 106 (1978)

Generation, distribution, and installation of cryptographic keys, *Matyas*, **17**, 2, 126 (1978)

Administrative control of computing service, *Gladney*, **17**, 2, 151 (1978)

Data processing spheres of control, *Davies*, **17**, 2, 179 (1978)

A time-sharing display terminal session manager, *McCrossin*, **17**, 3, 260 (1978)

Managing VM/CMS systems for user effectiveness, *Doherty*, **18**, 1, 143 (1979)

An integrated approach to centralized communications network management, *Weingarten*, **18**, 4, 484 (1979)

A sidestream approach using a small processor as a tool for managing communication systems, *Leach*, **19**, 1, 120 (1980)

Systems management, *Bird*, **19**, 1, 140 (1980)

The management of software engineering Part V: Software engineering management practices, *Quinnan*, **19**, 4, 466 (1980)

Processor, I/O path, and DASD configuration capacity, *Major*, **20**, 1, 63 (1981)

Interactive user productivity, *Thadhani*, **20**, 4, 407 (1981)

The VM/370 Resource Limiter, *Chess*, **20**, 4, 424 (1981)

Towards an integrated development environment, *Newman*, **21**, 1, 81 (1982)

Enterprise information analysis: Cost-benefit analysis and the data-managed system, *Parker*, **21**, 1, 108 (1982)

Management considerations for an Information Center, *Hammond*, **21**, 2, 131 (1982)

A case study of office workstation use, *Bullen*, **21**, 3, 351 (1982)

Analytic queuing model for CICS capacity planning, *Deitch*, **21**, 4, 454 (1982)

Communications Network Management enhancements for SNA networks: An overview, *Sullivan*, **22**, 1/2, 129 (1983)

An application of network management at a large computing service, *Garrigues*, **22**, 1/2, 143 (1983)

The Project Automated Librarian, *Prager*, **22**, 3, 214 (1983)

Performance and availability measurement of the IBM Information Network, *Bailey*, **22**, 4, 404 (1983)

Factors affecting programmer productivity during application development, *Thadhani*, **23**, 1, 19 (1984)

A comparative study of system response time on program developer productivity, *Lambert*, **23**, 1, 36 (1984)

Worldwide systems engineering, *Peck*, **24**, 3/4, 182 (1985)

Strategies for problem prevention, *Newton*, **24**, 3/4, 248 (1985)

An information technology architecture for change, *Mudie*, **24**, 3/4, 307 (1985)

YES/MVS and the automation of operations for large computer complexes, *Milliken*, **25**, 2, 159 (1986)

Computing as a tool for human augmentation, *Doherty*, **25**, 3/4, 306 (1986)

Networking

Evaluation of an interactive-batch system network, *Hobgood*, **11**, 1, 2 (1972)

Protocol for a computer network, *McKay*, **12**, 1, 94 (1973)

Describing data in computer networks, *Fredericksen*, **12**, 3, 257 (1973)

Systems Network Architecture: An overview, *McFadyen,* **15,** 1, 4 (1976)

The transmission subsystem in Systems Network Architecture, *Cullum,* **15,** 1, 24 (1976)

The role of the Network Control Program in Systems Network Architecture, *Hobgood,* **15,** 1, 39 (1976)

The Virtual Telecommunications Access Method: A Systems Network Architecture perspective, *Albrecht,* **15,** 1, 53 (1976)

Experiments in line quality monitoring, *Bryant,* **15,** 2, 124 (1976)

CICS/VS and its role in Systems Network Architecture, *Eade,* **16,** 3, 258 (1977)

Cryptography architecture for information security, *Lennon,* **17,** 2, 138 (1978)

Job networking, *Crabtree,* **17,** 3, 206 (1978)

Network job entry facility for JES2, *Simpson,* **17,** 3, 221 (1978)

Enhanced problem determination capability for teleprocessing, *Ford,* **17,** 3, 276 (1978)

Evolution of a virtual machine subsystem, *Hendricks,* **18,** 1, 111 (1979)

An introduction to network architectures and protocols, *Green,* **18,** 2, 202 (1979)

Public data networks: Their evolution, interfaces, and status, *Halsey,* **18,** 2, 223 (1979)

SNA and emerging international standards, *Corr,* **18,** 2, 244 (1979)

SNA multiple-system networking, *Gray,* **18,** 2, 263 (1979)

Routing and flow control in Systems Network Architecture, *Ahuja,* **18,** 2, 298 (1979)

Evolution of a laboratory communication network, *Moore,* **18,** 2, 315 (1979)

Potential technology implications for computers and telecommunications in the 1980s, *Frazer,* **18,** 2, 333 (1979)

Performance analysis of complex communications systems, *Stewart,* **18,** 3, 356 (1979)

An integrated approach to centralized communications network management, *Weingarten,* **18,** 4, 484 (1979)

A sidestream approach using a small processor as a tool for managing communication systems, *Leach,* **19,** 1, 120 (1980)

Distributed processing communications software support for operation within an SNA network, *Harrison,* **19,** 2, 192 (1980)

SNA flow control: Architecture and implementation, *George,* **21,** 2, 179 (1982)

X.25 and related recommendations in IBM products, *Deaton,* **22,** 1/2, 11 (1983)

Teletex–A worldwide link among office systems for electronic document exchange, *Moore,* **22,** 1/2, 30 (1983)

A token-ring network for local data communications, *Dixon,* **22,** 1/2, 47 (1983)

Reflections on VM/Pass-Through: A facility for interactive networking, *Mendelsohn,* **22,** 1/2, 63 (1983)

Communications Network Management enhancements for SNA networks: An overview, *Sullivan,* **22,** 1/2, 129 (1983)

An application of network management at a large computing service, *Garrigues,* **22,** 1/2, 143 (1983)

Advanced program-to-program communication in SNA, *Gray,* **22,** 4, 298 (1983)

SNA Distribution Services, *Housel,* **22,** 4, 319 (1983)

Interconnecting SNA networks, *Benjamin,* **22,** 4, 344 (1983)

An experimental address space isolation technique for SNA networks, *Ryder,* **22,** 4, 367 (1983)

Logical problem determination for SNA networks, *Weingarten,* **22,** 4, 387 (1983)

Performance and availability measurement of the IBM Information Network, *Bailey,* **22,** 4, 404 (1983)

SNA routing: Past, present, and possible future, *Jaffe,* **22,** 4, 417 (1983)

Defining routing tables for SNA networks, *Maruyama,* **22,** 4, 435 (1983)

Windows in the sky–Flow control in SNA networks with satellite links, *Grover,* **22,** 4, 451 (1983)

Performance issues in local-area networks, *Bux,* **23,** 4, 351 (1984)

The System Usability Process for Network Management Products, *Gottschalk,* **25,** 1, 83 (1986)

Network management software usability test design and implementation, *Percival,* **25,** 1, 92 (1986)

Open Systems Interconnection, *Aschenbrenner,* **25,** 3/4, 369 (1986)

Structures for networks of systems, *Scherr,* **26,** 1, 4 (1987)

SNA: Current requirements and direction, *Sundstrom,* **26,** 1, 13 (1987)

Prospects and design choices for integrated private networks, *Green,* **26,** 1, 37 (1987)

A large-scale computer conferencing system, *Chess,* **26,** 1, 138 (1987)

A perspective on Advanced Peer-to-Peer Networking, *Green,* **26,** 4, 414 (1987)

Implementing System/36 Advanced Peer-to-Peer Networking, *Sultan,* **26,** 4, 429 (1987)

SNA network management directions, *Rose,* **27,** 1, 3 (1988)

Utilizing the SNA Alert in the management of multivendor networks, *Moore,* **27,** 1, 15 (1988)

NetView/PC, *Ahmadi,* **27,** 1, 32 (1988)

An integrated network management product, *Kanyuh,* **27,** 1, 45 (1988)

An architecture for a business and information system, *Devlin,* **27,** 1, 60 (1988)

Common Communications Support in Systems Application Architecture, *Ahuja,* **27,** 3, 264 (1988)

SAA distributed processing, *Scherr,* **27,** 3, 370 (1988)

Office Applications

A guide to financial planning tools and techniques, *Dzielinski,* **12,** 2, 126 (1973)

An office communications system, *Engel,* **18,** 3, 402 (1979)

A research perspective on computer-assisted office work, *Gruhn,* **18,** 3, 432 (1979)

Experimental page makeup of text with graphics on a raster printer, *Shepherd,* **19,** 3, 345 (1980)

Electronic information interchange in an office environment, *DeSousa,* **20,** 1, 4 (1981)

User-definable software applied to a real-time ambient air quality monitoring system, *Halpern,* **20,** 1, 86 (1981)

A system for the automated office environment, *Gardner,* **20,** 3, 321 (1981)

JANUS: An interactive document formatter based on declarative tags, *Chamberlin,* **21,** 3, 250 (1982)

Office-by-Example: A business language that unifies data and word processing and electronic mail, *Zloof,* **21,** 3, 272 (1982)

The EPISTLE text-critiquing system, *Heidorn,* **21,** 3, 305 (1982)

OPAS: An office procedure automation system, *Lum,* **21,** 3, 327 (1982)

Teletex–A worldwide link among office systems for electronic document exchange, *Moore,* **22,** 1/2, 30 (1983)

Speech filing–An office system for principals, *Gould,* **23,** 1, 65 (1984)

Use of images in commercial and office systems, *Somerville,* **23,** 3, 281 (1984)

Operating Systems

Virtual storage and virtual machine concepts, *Parmelee,* **11,** 2, 99 (1972)

Virtual machine computing in an engineering environment, *McGrath,* **11,** 2, 131 (1972)

Uses of virtual storage systems in a scientific environment, *Callaway*, **11**, 3, 200 (1972)

User program performance in virtual storage systems, *Morrison*, **12**, 3, 216 (1973)

Functional structure of IBM virtual storage operating systems Part I: Influences of dynamic address translation on operating system technology, *Auslander*, **12**, 4, 368 (1973)

Functional structure of IBM virtual storage operating systems Part II: OS/VS2-2 concepts and philosophies, *Scherr*, **12**, 4, 382 (1973)

Functional structure of IBM virtual storage operating systems Part III: Architecture and design of DOS/VS, *Birch*, **12**, 4, 401 (1973)

VSAM data set design parameters, *Keehn*, **13**, 3, 186 (1974)

OS/VS1 concepts and philosophies, *Wheeler*, **13**, 3, 213 (1974)

Operating system integrity in OS/VS2, *McPhee*, **13**, 3, 230 (1974)

The job entry subsystem of OS/VS1, *Baily*, **13**, 3, 253 (1974)

The OS/VS2 Release 2 System Resources Manager, *Lynch*, **13**, 4, 274 (1974)

Performance measurement tools for VM/370, *Callaway*, **14**, 2, 134 (1975)

Hierarchical approach to computer system integrity, *Donovan*, **14**, 2, 188 (1975)

An access control mechanism for computing resources, *Gladney*, **14**, 3, 212 (1975)

Tuning a virtual storage system, *Anderson*, **14**, 3, 246 (1975)

Performance analysis of virtual memory time-sharing systems, *Bard*, **14**, 4, 366 (1975)

The Virtual Telecommunications Access Method: A Systems Network Architecture perspective, *Albrecht*, **15**, 1, 53 (1976)

LABS/7–a distributed real-time operating system, *Raimondi*, **15**, 1, 81 (1976)

Penetrating an operating system: a study of VM/370 integrity, *Attanasio*, **15**, 1, 102 (1976)

A model of large program development, *Belady*, **15**, 3, 225 (1976)

MARC: MVS archival storage and recovery program, *Considine*, **16**, 4, 378 (1977)

Network job entry facility for JES2, *Simpson*, **17**, 3, 221 (1978)

Performance tuning in OS/VS2 MVS, *Beretvas*, **17**, 3, 290 (1978)

Performance investigations with a DOS/VS-based operating system model, *Kraemer*, **17**, 4, 409 (1978)

A performance model of MVS, *Chiu*, **17**, 4, 444 (1978)

VM/370–a study of multiplicity and usefulness, *Seawright*, **18**, 1, 4 (1979)

The changing virtual machine environment: Interfaces to real hardware, virtual hardware, and other virtual machines, *MacKinnon*, **18**, 1, 18 (1979)

VM/370 asymmetric multiprocessing, *Holley*, **18**, 1, 47 (1979)

A formal approach for communication between logically isolated virtual machines, *Jensen*, **18**, 1, 71 (1979)

Virtual Control Storage–security measures in VM/370, *Attanasio*, **18**, 1, 93 (1979)

Evolution of a virtual machine subsystem, *Hendricks*, **18**, 1, 111 (1979)

Managing VM/CMS systems for user effectiveness, *Doherty*, **18**, 1, 143 (1979)

State sampling of interactive VM/370 users, *Tetzlaff*, **18**, 1, 164 (1979)

An operating system for distributed processing–DPPX, *Kiely*, **18**, 4, 507 (1979)

I/O facilities of the Distributed Processing Programming Executive (DPPX), *Albrecht*, **18**, 4, 526 (1979)

Data Management for the Distributed Processing Programming Executive (DPPX), *Fitzgerald*, **18**, 4, 547 (1979)

An MVS tuning approach, *Schardt*, **19**, 1, 102 (1980)

Analysis of free-storage algorithms–revisited, *Bozman*, **23**, 1, 44 (1984)

VM/370, Attached Processor, and multiprocessor performance study, *Tetzlaff*, **23**, 4, 375 (1984)

Expanded personal computing power and capability, *Korn*, **24**, 1, 26 (1985)

A page-swapping prototype for VM/HPO, *Tetzlaff*, **26**, 2, 215 (1987)

Advanced Interactive Executive (AIX) operating system overview, *Loucks*, **26**, 4, 326 (1987)

Advanced Interactive Executive program development environment, *Cordell*, **26**, 4, 361 (1987)

AIX usability enhancements and human factors, *Waters*, **26**, 4, 383 (1987)

The design of Operating System/2, *Kogan*, **27**, 2, 90 (1988)

Writing an Operating System/2 application, *Cook*, **27**, 2, 134 (1988)

Understanding device drivers in Operating System/2, *Mizell*, **27**, 2, 170 (1988)

The Cross System Product application generator: An evolution, *Haynes*, **27**, 3, 384 (1988)

Performance

Evaluation of an interactive-batch system network, *Hobgood*, **11**, 1, 2 (1972)

Queuing simulation using a random number generator, *Rechtschaffen*, **11**, 3, 255 (1972)

Techniques for developing analytic models, *Anthony*, **11**, 4, 316 (1972)

User program performance in virtual storage systems, *Morrison*, **12**, 3, 216 (1973)

Experimental evaluation of system performance, *Bard*, **12**, 3, 302 (1973)

Performance analysis for the Skylab terminal system, *Mancini*, **13**, 2, 94 (1974)

VSAM data set design parameters, *Keehn*, **13**, 3, 186 (1974)

Elements of probability for system design, *Allen*, **13**, 4, 325 (1974)

Store performance studies for the Supermarket System, *Metz*, **14**, 1, 46 (1975)

Design and performance considerations for the Retail Store System, *Berk*, **14**, 1, 64 (1975)

Performance measurement tools for VM/370, *Callaway*, **14**, 2, 134 (1975)

Elements of queuing theory for system design, *Allen*, **14**, 2, 161 (1975)

Tuning a virtual storage system, *Anderson*, **14**, 3, 246 (1975)

Evaluating system changes under uncontrolled workloads: a case study, *Friedman*, **14**, 4, 340 (1975)

Performance analysis of virtual memory time-sharing systems, *Bard*, **14**, 4, 366 (1975)

Interactive modeling of computer systems, *Reiser*, **15**, 4, 309 (1976)

Performance tuning in OS/VS2 MVS, *Beretvas*, **17**, 3, 290 (1978)

Performance investigations with a DOS/VS-based operating system model, *Kraemer*, **17**, 4, 409 (1978)

A performance model of MVS, *Chiu*, **17**, 4, 444 (1978)

Performance analysis of complex communications systems, *Stewart*, **18**, 3, 356 (1979)

System capacity and performance evaluation, *Schiller*, **19**, 1, 46 (1980)

Modeling considerations for predicting performance of CICS/VS systems, *Seaman*, **19**, 1, 68 (1980)

The role of detailed simulation in capacity planning, *Nguyen*, **19**, 1, 81 (1980)

An MVS tuning approach, *Schardt*, **19**, 1, 102 (1980)

System contention analysis–An alternate approach to system tuning, *Yuval,* **19,** 2, 208 (1980)

Capacity analysis of the Mass Storage System, *Misra,* **20,** 3, 346 (1981)

Design and use of a program execution analyzer, *Power,* **22,** 3, 271 (1983)

IBM Database 2 performance: Design, implementation, and tuning, *Cheng,* **23,** 2, 189 (1984)

Managing IBM Database 2 buffers to maximize performance, *Teng,* **23,** 2, 211 (1984)

Performance issues in local-area networks, *Bux,* **23,** 4, 351 (1984)

VM/370, Attached Processor, and multiprocessor performance study, *Tetzlaff,* **23,** 4, 375 (1984)

Cache-DASD storage design for improving system performance, *Grossman,* **24,** 3/4, 316 (1985)

IBM 3090 performance: A balanced system approach, *Singh,* **25,** 1, 20 (1986)

Engineering and scientific processing on the IBM 3090, *Gibson,* **25,** 1, 36 (1986)

Vector system performance of the IBM 3090, *Clark,* **25,** 1, 63 (1986)

Program locality of vectorized applications running on the IBM 3090 with Vector Facility, *So,* **27,** 4, 436 (1988)

Programming

Readings in microprogramming, *Davies,* **11,** 1, 16 (1972)

Chief programmer team management of production programming, *Baker,* **11,** 1, 56 (1972)

A guide to programming tools and techniques, *Pomeroy,* **11,** 3, 234 (1972)

User program performance in virtual storage systems, *Morrison,* **12,** 3, 216 (1973)

Design of a checkout compiler, *Marks,* **12,** 3, 315 (1973)

Design of tightly-coupled multiprocessing programming, *Arnold,* **13,** 1, 60 (1974)

Structured design, *Stevens,* **13,** 2, 115 (1974)

A program generator, *Hagamen,* **14,** 2, 102 (1975)

Structured programming for virtual storage systems, *Rogers,* **14,** 4, 385 (1975)

HIPO and integrated program design, *Stay,* **15,** 2, 143 (1976)

Top-down development using a program design language, *Van Leer,* **15,** 2, 155 (1976)

Design and code inspections to reduce errors in program development, *Fagan,* **15,** 3, 182 (1976)

A model of large program development, *Belady,* **15,** 3, 225 (1976)

A method of programming measurement and estimation, *Walston,* **16,** 1, 54 (1977)

IBM's Santa Teresa Laboratory–Architectural design for program development, *McCue,* **17,** 1, 4 (1978)

A method for the time analysis of programs, *de Freitas,* **17,** 1, 26 (1978)

Measuring programming quality and productivity, *Jones,* **17,** 1, 39 (1978)

The Extended Control Language of MPSX/370 and possible applications, *Slate,* **17,** 1, 64 (1978)

Solving the installation scheduling problem using mixed integer linear programming, *Chen,* **17,** 1, 82 (1978)

Data Stream Linkage Mechanism, *Morrison,* **17,** 4, 383 (1978)

Automatic programming for energy management using sensor based computers, *Shah,* **18,** 3, 457 (1979)

The management of software engineering Part I: Principles of software engineering, *Mills,* **19,** 4, 414 (1980)

The management of software engineering Part II: Software engineering program, *O'Neill,* **19,** 4, 421 (1980)

The management of software engineering Part III: Software design practices, *Linger,* **19,** 4, 432 (1980)

The management of software engineering Part IV: Software development practices, *Dyer,* **19,** 4, 451 (1980)

The management of software engineering Part V: Software engineering management practices, *Quinnan,* **19,** 4, 466 (1980)

Application development system: The software architecture of the IBM Health Care Support/DL/I-Patient Care System, *Mishelevich,* **19,** 4, 478 (1980)

A system for constructing linear programming models, *Katz,* **19,** 4, 505 (1980)

The Modular Application Customizing System, *Gordon,* **19,** 4, 521 (1980)

GREENPRINT: A graphic representation of structured programs, *Belady,* **19,** 4, 542 (1980)

User-definable software applied to a real-time ambient air quality monitoring system, *Halpern,* **20,** 1, 86 (1981)

Natural language programming: Styles, strategies, and contrasts, *Miller,* **20,** 2, 184 (1981)

A perspective on software science, *Christensen,* **20,** 4, 372 (1981)

System Productivity Facility, *Joslin,* **20,** 4, 388 (1981)

Business Systems Planning and Business Information Control Study: A comparison, *Zachman,* **21,** 1, 31 (1982)

Towards an integrated development environment, *Newman,* **21,** 1, 81 (1982)

How data flow can improve application development productivity, *Stevens,* **21,** 2, 162 (1982)

Technique for assessing external design of software, *Pearsall,* **21,** 2, 211 (1982)

Abstract design and program translator: New tools for software design, *Archibald,* **22,** 3, 170 (1983)

The system architecture of EAS-E: An integrated programming and data base language, *Pazel,* **22,** 3, 188 (1983)

A simple architecture for consistent application program design, *Rogers,* **22,** 3, 199 (1983)

The Project Automated Librarian, *Prager,* **22,** 3, 214 (1983)

Automatic generation of random self-checking test cases, *Bird,* **22,** 3, 229 (1983)

Full-screen testing of interactive applications, *Maurer,* **22,** 3, 246 (1983)

Software reliability analysis, *Misra,* **22,** 3, 262 (1983)

Design and use of a program execution analyzer, *Power,* **22,** 3, 271 (1983)

Architecture prototyping in the software engineering environment, *Beregi,* **23,** 1, 4 (1984)

Factors affecting programmer productivity during application development, *Thadhani,* **23,** 1, 19 (1984)

A comparative study of system response time on program developer productivity, *Lambert,* **23,** 1, 36 (1984)

Playback: A method for evaluating the usability of software and its documentation, *Neal,* **23,** 1, 82 (1984)

The design of the REXX language, *Cowlishaw,* **23,** 4, 326 (1984)

The IBM large-systems software development process: Objectives and direction, *Humphrey,* **24,** 2, 76 (1985)

A programming process architecture, *Radice,* **24,** 2, 79 (1985)

A programming process study, *Radice,* **24,** 2, 91 (1985)

Automating the software development process, *Hoffnagle,* **24,** 2, 102 (1985)

Quality emphasis at IBM's Software Engineering Institute, *Carpenter,* **24,** 2, 121 (1985)

PDM: A requirements methodology for software system enhancements, *Mays,* **24,** 2, 134 (1985)

A process-integrated approach to defect prevention, *Jones,* **24,** 2, 150 (1985)

Programming process productivity measurement system for System/370, *Flaherty,* **24,** 2, 168 (1985)

Software engineering: An emerging discipline, *Goldberg,* **25,** 3/4, 334 (1986)

An incidence-matrix-driven panel system for the IBM PC, *Halpern,* **26,** 2, 201 (1987)

Advanced Interactive Executive program development environment, *Cordell*, **26**, 4, 361 (1987)

Box structured information systems, *Mills*, **26**, 4, 395 (1987)

COBOL/2: The next generation in applications programming, *Sales*, **27**, 2, 158 (1988)

Application enabling in SAA, *Wolford*, **27**, 3, 301 (1988)

Enabling the user interface, *Uhlir*, **27**, 3, 306 (1988)

Distributed database for SAA, *Reinsch*, **27**, 3, 362 (1988)

The Cross System Product application generator: An evolution, *Haynes*, **27**, 3, 384 (1988)

IBM's directions in technical computing, *Prairie*, **27**, 4, 393 (1988)

Engineering and Scientific Subroutine Library for the IBM 3090 Vector Facility, *McComb*, **27**, 4, 404 (1988)

Programming style on the IBM 3090 Vector Facility considering both performance and flexibility, *Samukawa*, **27**, 4, 453 (1988)

Effective utilization of IBM 3090 large virtual storage in the numerically intensive computations of *ab initio* molecular orbitals, *Sakaki*, **27**, 4, 528 (1988)

Personal Systems

Directions in cooperative processing between workstations and hosts, *Goldstein*, **23**, 3, 236 (1984)

System/370 capability in a desktop computer, *Kozuh*, **23**, 3, 245 (1984)

A tight coupling of workstations, *Chess*, **23**, 3, 255 (1984)

Architecture implications in the design of microprocessors, *Matick*, **23**, 3, 264 (1984)

Security considerations for personal computers, *Murray*, **23**, 3, 297 (1984)

Standardized graphics on the IBM Personal Computer, *Clarkson*, **24**, 1, 3 (1985)

A professional graphics controller, *Duke*, **24**, 1, 14 (1985)

Expanded personal computing power and capability, *Korn*, **24**, 1, 26 (1985)

The C programming language and a C compiler, *Ryan*, **24**, 1, 37 (1985)

Design considerations for IBM Personal Computer Professional FORTRAN, an optimizing compiler, *Roberts*, **24**, 1, 49 (1985)

An APL system for the IBM Personal Computer, *Tavera*, **24**, 1, 61 (1985)

A single-system interface using the IBM 3270-PC, *Ghiotti*, **24**, 3/4, 236 (1985)

Workstations and mainframe computers working together, *Kravitz*, **25**, 1, 116 (1986)

IBM small-system architecture and design–Past, present, and future, *Henry*, **25**, 3/4, 321 (1986)

The design of Operating System/2, *Kogan*, **27**, 2, 90 (1988)

OS/2 Query Manager overview and prompted interface, *Watson*, **27**, 2, 119 (1988)

Writing an Operating System/2 application, *Cook*, **27**, 2, 134 (1988)

COBOL/2: The next generation in applications programming, *Sales*, **27**, 2, 158 (1988)

Understanding device drivers in Operating System/2, *Mizell*, **27**, 2, 170 (1988)

VGA–Design choices for a new video subsystem, *Thompson*, **27**, 2, 185 (1988)

The Realtime Interface Co-Processor Multiport/2 adapter, *Sykes*, **27**, 2, 198 (1988)

Reliability and Availability

Reliability, availability, and serviceability design considerations for the Supermarket and Retail Store Systems, *Hippert*, **14**, 1, 81 (1975)

Experiments in line quality monitoring, *Bryant*, **15**, 2, 124 (1976)

Logical problem determination for SNA networks, *Weingarten*, **22**, 4, 387 (1983)

Performance and availability measurement of the IBM Information Network, *Bailey*, **22**, 4, 404 (1983)

Performance considerations for a distributed data processing system designed for high availability, *Agassi*, **24**, 3/4, 200 (1985)

An approach to high availability in high-transaction-rate systems, *Brooks*, **24**, 3/4, 279 (1985)

Improving availability of software subsystems through on-line error detection, *Koved*, **25**, 1, 105 (1986)

System Design

Design features of a real-time check-clearing system, *Banham*, **11**, 4, 329 (1972)

Advanced function extended with tightly-coupled multiprocessing, *MacKinnon*, **13**, 1, 32 (1974)

Performance analysis for the Skylab terminal system, *Mancini*, **13**, 2, 94 (1974)

System/7 in a hierarchical laboratory automation system, *Cole*, **13**, 4, 307 (1974)

Improving system usability for business professionals, *Helander*, **20**, 3, 294 (1981)

The design rationale of the System/38 user interface, *Botterill*, **21**, 4, 384 (1982)

Ease of use: A system design challenge, *Branscomb*, **23**, 3, 224 (1984)

System/370 capability in a desktop computer, *Kozuh*, **23**, 3, 245 (1984)

The IBM 3090 system: An overview, *Tucker*, **25**, 1, 4 (1986)

Engineering and scientific processing on the IBM 3090, *Gibson*, **25**, 1, 36 (1986)

The structure of System/88, a fault-tolerant computer, *Harrison*, **26**, 3, 293 (1987)

The IBM RT PC ROMP processor and memory management unit architecture, *Simpson*, **26**, 4, 346 (1987)

Box structured information systems, *Mills*, **26**, 4, 395 (1987)

The design of Operating System/2, *Kogan*, **27**, 2, 90 (1988)

Understanding device drivers in Operating System/2, *Mizell*, **27**, 2, 170 (1988)

Common User Access–A consistent and usable human-computer interface for the SAA environments, *Berry*, **27**, 3, 281 (1988)

Integrating applications with SAA, *Buchwald*, **27**, 3, 315 (1988)

Designing SAA applications and user interfaces, *Dunfee*, **27**, 3, 325 (1988)

Program locality of vectorized applications running on the IBM 3090 with Vector Facility, *So*, **27**, 4, 436 (1988)

ICAP 3090: Parallel processing for large-scale scientific and engineering problems, *Clementi*, **27**, 4, 475 (1988)

Security

Hierarchical approach to computer system integrity, *Donovan*, **14**, 2, 188 (1975)

An access control mechanism for computing resources, *Gladney*, **14**, 3, 212 (1975)

Generalized audit trail requirements and concepts for data base applications, *Bjork*, **14**, 3, 229 (1975)

Penetrating an operating system: a study of VM/370 integrity, *Attanasio*, **15**, 1, 102 (1976)

A cryptographic key management scheme for implementing the Data Encryption Standard, *Ehrsam*, **17**, 2, 106 (1978)

Generation, distribution, and installation of cryptographic keys, *Matyas*, **17**, 2, 126 (1978)

Cryptography architecture for information security, *Lennon*, **17**, 2, 138 (1978)

Virtual Control Storage–security measures in VM/370, *Attanasio,* **18,** 1, 93 (1979)

Data base security: requirements, policies, and models, *Wood,* **19,** 2, 229 (1980)

The IPS cryptographic programs, *Konheim,* **19,** 2, 253 (1980)

Security considerations for personal computers, *Murray,* **23,** 3, 297 (1984)

An overview of computer security, *Summers,* **23,** 4, 309 (1984)

Simulation

A general management business simulation in APL, *Wahi,* **11,** 2, 169 (1972)

Channel and direct access device architecture, *Brown,* **11,** 3, 186 (1972)

Queuing simulation using a random number generator, *Rechtschaffen,* **11,** 3, 255 (1972)

Techniques for developing analytic models, *Anthony,* **11,** 4, 316 (1972)

Concepts of financial models, *Kingston,* **12,** 2, 113 (1973)

Financial modeling on small systems, *Gordon,* **12,** 2, 161 (1973)

Interactive simulation for banking, *Brown,* **12,** 2, 172 (1973)

Direct-access device simulation, *Nahouraii,* **13,** 1, 19 (1974)

A model for the evaluation of storage hierarchies, *Gecsei,* **13,** 2, 163 (1974)

Store performance studies for the Supermarket System, *Metz,* **14,** 1, 46 (1975)

Design and performance considerations for the Retail Store System, *Berk,* **14,** 1, 64 (1975)

Productivity of computer-dependent workers, *Streeter,* **14,** 3, 292 (1975)

A model of large program development, *Belady,* **15,** 3, 225 (1976)

Interactive modeling of computer systems, *Reiser,* **15,** 4, 309 (1976)

Storage and access in relational data bases, *Blasgen,* **16,** 4, 363 (1977)

An input-output econometric model, *Sarma,* **16,** 4, 398 (1977)

Performance investigations with a DOS/VS-based operating system model, *Kraemer,* **17,** 4, 409 (1978)

A performance model of MVS, *Chiu,* **17,** 4, 444 (1978)

Performance analysis of complex communications systems, *Stewart,* **18,** 3, 356 (1979)

A distributed information system study, *Ziegler,* **18,** 3, 374 (1979)

Modeling considerations for predicting performance of CICS/VS systems, *Seaman,* **19,** 1, 68 (1980)

The role of detailed simulation in capacity planning, *Nguyen,* **19,** 1, 81 (1980)

A system for constructing linear programming models, *Katz,* **19,** 4, 505 (1980)

Software simulation as a tool for usable product design, *Clark,* **20,** 3, 272 (1981)

Analytic queuing model for CICS capacity planning, *Deitch,* **21,** 4, 454 (1982)

Modeling distributed processing across multiple CICS/VS sites, *Acker,* **21,** 4, 471 (1982)

The System Planning Grid: A model for building integrated information systems, *Buckelew,* **24,** 3/4, 294 (1985)

Standards

A cryptographic key management scheme for implementing the Data Encryption Standard, *Ehrsam,* **17,** 2, 106 (1978)

Public data networks: Their evolution, interfaces, and status, *Halsey,* **18,** 2, 223 (1979)

SNA and emerging international standards, *Corr,* **18,** 2, 244 (1979)

X.25 and related recommendations in IBM products, *Deaton,* **22,** 1/2, 11 (1983)

Teletex–A worldwide link among office systems for electronic document exchange, *Moore,* **22,** 1/2, 30 (1983)

Open Systems Interconnection, *Aschenbrenner,* **25,** 3/4, 369 (1986)

Storage Systems

Virtual storage and virtual machine concepts, *Parmelee,* **11,** 2, 99 (1972)

Encoding verbal information as unique numbers, *Hagamen,* **11,** 4, 278 (1972)

User program performance in virtual storage systems, *Morrison,* **12,** 3, 216 (1973)

Indexing design considerations, *Wagner,* **12,** 4, 351 (1973)

Optimizing program placement in virtual systems, *Ryder,* **13,** 4, 292 (1974)

MARC: MVS archival storage and recovery program, *Considine,* **16,** 4, 378 (1977)

National Westminster Bank mass storage archiving, *Gravina,* **17,** 4, 344 (1978)

Potential technology implications for computers and telecommunications in the 1980s, *Frazer,* **18,** 2, 333 (1979)

Capacity analysis of the Mass Storage System, *Misra,* **20,** 3, 346 (1981)

Analysis of free-storage algorithms–revisited, *Bozman,* **23,** 1, 44 (1984)

Cache-DASD storage design for improving system performance, *Grossman,* **24,** 3/4, 316 (1985)

Vector system performance of the IBM 3090, *Clark,* **25,** 1, 63 (1986)

Impact of memory systems on computer architecture and system organization, *Matick,* **25,** 3/4, 274 (1986)

Testing

Testing in a complex systems environment, *Duke,* **14,** 4, 353 (1975)

Automatic generation of random self-checking test cases, *Bird,* **22,** 3, 229 (1983)

Full-screen testing of interactive applications, *Maurer,* **22,** 3, 246 (1983)

Strategies for problem prevention, *Newton,* **24,** 3/4, 248 (1985)

Network management software usability test design and implementation, *Percival,* **25,** 1, 92 (1986)

Improving availability of software subsystems through on-line error detection, *Koved,* **25,** 1, 105 (1986)

IBM Systems Journal Abstracts 1981–1988

Volume 20, Number 1, 1981 *G321-0064*

Electronic information interchange in an office environment *by M. R. DeSousa, p. 4.* This paper describes an architectural approach that provides information interchange across a broad spectrum of user applications and office automation offerings. Some of the architectures described herein are currently implemented in existing IBM products. These and other architectures will provide the basis for document interchange capability between products such as the IBM 5520 Administrative System, the IBM System/370 Distributed Office Support System (DISOSS), and the IBM Displaywriter System. Specifically described is a document distribution architecture and its associated data streams. Transforms can be utilized to interchange between these data streams and others.

A general overview of the architectures as opposed to a detailed technical description is provided. The architectures described are protocols for interchange between application processes; they do not address the specific user interface. The document distribution architectures utilize SNA for data transmission and communications control facilities.

A primer on relational data base concepts *by G. Sandberg, p. 23.* Basic concepts of relational data base management systems are described. Characteristics of the relational approach are identified and compared with present implementations of hierarchical and network data base systems. Depending on the application, a user may experience one or more of the following benefits of relational systems described in this paper: ease of understanding, increased data independence, ease of use, sound theoretical basis, and generalized data definition. Types of applications most suited to hierarchical and network data base systems are also compared and contrasted.

System R: An architectural overview *by M. W. Blasgen, M. M. Astrahan, D. D. Chamberlin, J. N. Gray, W. F. King, B. G. Lindsay, R. A. Lorie, J. W. Mehl, T. G. Price, G. R. Putzolu, M. Schkolnick, P. G. Selinger, D. R. Slutz, H. R. Strong, I. L. Traiger, B. W. Wade, and R. A. Yost, p. 41.* System R is an experimental data base management system that was designed to be unusually easy to use. System R supports a high-level relational user language called SQL, which may be used by ad hoc users at terminals or by programmers as an imbedded data sublanguage in PL/I or COBOL. This paper describes the overall architecture of the system, including the Relational Data System (RDS) and the Research Storage System (RSS).

RDS is a data base language compiler. Host language programs with imbedded SQL statements are compiled by System R, which replaces the SQL statements with calls to a machine-language access module. The compilation approach removes much of the work of parsing, name binding, and optimization from the path of a running program, enabling highly efficient support for repetitive transactions. In contrast, the RSS is a low-level DBMS, supporting simple record-at-a-time operators, but with rather sophisticated transaction management, recovery, and concurrency control.

Processor, I/O path, and DASD configuration capacity *by J. B. Major, p. 63.* This paper extends a particular capacity planning approach to include usage accounting by business element of Input/Output path and Direct Access Storage Device resources. A simple non-linear procedure is outlined to size host configurations that can process workloads at specified rates. An important feature of the procedure is to take account of a law of diminishing returns, which is that doubling the number of components does not double the amount of work done. Discussed are configuration relationships involving TSO and DB/DC subsystem sizing, tuning, workload variability, data considerations, and hardware and software considerations. Typical, but hypothetical, examples are presented.

User-definable software applied to a real-time ambient air quality monitoring system *by P. Halpern and J. W. Rettberg, p. 86.* With increased tightening of air quality

regulations, more systems for monitoring air quality became necessary. The greater use of such systems further stimulated the development of sensor-based systems that require less programming effort. The goal was to have a practical, real-time system with the ability to change the configuration of the sensors without extensively modifying the associated software. Described is a prototypical system that avoids the necessity of reprogramming every time sensors are changed.

Volume 20, Number 2, 1981 *G321-0065*

Procedures of the Human Factors Center at San Jose *by R. S. Hirsch, p. 123.* The work performed at the Human Factors Center located at IBM's development facility in San Jose, California, is representative of human factors work being done by groups of human factors specialists throughout IBM. A few of the projects that the Center was involved with are described as examples to show how human factors concerns are studied in the development of products and systems. The examples were selected to indicate the broad nature of the problems studied and include hardware and software areas. The complete scientific techniques used in the projects are not discussed in this paper so that the focus of discussion will be on the nature, scope, and methodology of the human factors work. The computing and data collection systems used for human factors tests are briefly discussed.

Effects of manual style on performance in education and machine maintenance *by J. M. Judisch, B. A. Rupp, and R. A. Dassinger, p. 172.* Discussed is a study of human factors that was designed to measure the time to perform maintenance using two types of manuals (format) presented in two media types, for a total of four conditions. The types of manuals were the then-current Field Engineering Maintenance Manual (FEMM) and Field Engineering Theory of Operation Manual (FETON) in both hard copy and microfiche, and a new Graphic Integrated Manual (GIM) covering the same subject matter, also in hard copy and microfiche. The objective of the study was to compare performance in solving problems on an electromechanical machine, the IBM 5424 Multi-Function Card Unit, through the use of standard and graphic integrated manuals in both hard copy and microfiche for that machine. Test results are analyzed and conclusions are presented. The general conclusion is that the new graphic integrated manuals in hard copy format lead to better performance both in education and on the job.

Natural language programming: Styles, strategies, and contrasts *by L. A. Miller, p. 184.* College students who were not familiar with computers were asked to produce written natural language procedural instructions as directions for others to follow. These directions were solutions for six file-manipulation problems that also could reasonably be solved by writing computer programs. The written texts were examined from five points of view: solution correctness, preferences of expression, contextual referencing, word usage, and formal programming languages. The results provide insight both on the manner in which people express computer-like procedures "naturally" and on what features programming languages should include if they are to be more "natural-like."

Human factors in the development of a family of plant data communication terminals *by M. Ominsky, p. 216.* Developing a set of terminals for users who had no computer experience and whose normal jobs could not be subject to interference involved human factors. Most of the design work focused on the keyboard and display interfaces of the terminals. Studies were made, alternative designs were considered, and tests were performed to ensure that the equipment was easy to use and provided acceptable speed and accuracy.

Human factors in communication *by J. C. Thomas and J. M. Carroll, p. 237.* One way of conceptualizing many of the human factors issues in interactive computing is as issues in communication about computers. Presented are a framework for this conceptualization and a review of research addressed to several levels of the communication process, Communication as an ill-structured design process is analyzed and contrasted with a process of algorithmic encoding and decoding. The design framework is then applied to examinations of how people name and refer to entities, how people understand and express relations (quantifiers and other predicates) between entities, how more complex communications (business letters) are created, and how preprinted forms reflect previous knowledge.

Volume 20, Number 3, 1981 *G321-0066*

Software simulation as a tool for usable product design *by I. A. Clark, p. 272.* A design exercise performed by human factors specialists is described. In this exercise a front-of-screen simulation of the Interactive Chart Utility was written before a working prototype was available in order to draft and test a series of on-line instructional (HELP) panels for incorporation into the final product. Trials were run in which the keyboard activity and utterances of naive subjects were recorded for later action replay, before and after redrafting the simulation. Three objective measures to detect the resulting improvement are considered, and the most robust identified.

Improving system usability for business professionals *by G. A. Helander, p. 294.* As businesses increase their emphasis on productivity, data processing departments face rising demand for computer services from people with no data processing training or background. To be effective, these services must be easy to use. This article discusses some usability considerations and how they were applied in developing an end user system. It relates experiences and observations in developing a system that is marketed in Canada under the name Interactive Extension Facilities. This system is an extension to VM/370 and CMS and was developed by the IBM Canada Limited Laboratory (Toronto) to enhance the ability of business professionals to do their work without becoming data processing specialists.

Improving the usability of programming publications *by F. J. Bethke, W. M. Dean, P. H. Kaiser, E. Ort, and F. H. Pessin, p. 306.* This paper summarizes the work of a study group on ways to improve the usability of publications that support programming products. Task orientation, an approach to providing, organizing, and packaging information, is covered, together with innovations to improve the usability of programming publications: ease-of-use education, measurement of user opinion, and incorporating usability into the publications development process.

A system for the automated office environment *by P. C. Gardner, Jr., p. 321.* A review of the history of an office system application is presented, highlighting the learning process that took place during its evolutionary development. This office system has served as the basis for a PRPQ (customized program) recently announced by IBM and known as the Professional Office System (PROFS). The general application architecture is discussed, with a specific focus on the use of virtual machines. Functional details of the various components are described, and the key distinction between office systems and office automation is addressed. The paper also discusses usage of the system and points up some of the benefits being realized by current users of a prototype system in IBM. The application function review details the electronic document distribution capabilities of the system.

Capacity analysis of the Mass Storage System *by P. N. Misra, p. 346.* Performance of the IBM 3850 Mass Storage System (MSS) is analyzed with a view toward workload planning. Simple analytical models are discussed. The notion of staging capacity of the MSS is defined and analyzed. The main result is a set of staging capacity curves that define the processing ability of the MSS to stage and destage data support concurrent execution of the user programs.

Volume 20, Number 4, 1981 *G321-0067*

A perspective on software science *by K. Christensen, G. P. Fitsos, and C. P. Smith, p. 372.* This paper provides an overview of a new approach to the measurement of software. The measurements are based on the count of operators and operands contained in a program. The measurement methodologies are consistent across programming language barriers. Practical significance is discussed, and areas are identified for additional research and validation.

System Productivity Facility *by P. H. Joslin, p. 388.* This paper discusses the purpose and design of a program called the System Productivity Facility (SPF). Perspective is provided by means of a brief summary of the earlier Structure Programming Facility (also termed SPF) and the requirements that led to a transformation of the earlier program into a new cross-system dialog manager. The new control facilities are explained to illustrate how the dialog manager supports a wide variety of interactive applications. Ways in which application development is simplified in the areas of data handling and display processing are explored. The purpose of the new table and file tailoring services is explained, and the error recovery philosophy is described.

Interactive user productivity *by A. J. Thadhani, p. 407.* Interactive user productivity is a measure of effective communication between man and the computer. Explored in this paper is the relationship between computer response time and user performance, and the separation of user cost from system cost. Strategies for effectively managing installations are presented and discussed.

The VM/370 Resource Limiter *by D. M. Chess and G. Waldbaum, p. 424.* The VM/370 Resource Limiter (RESLIM), a facility available on the computer systems of the IBM Thomas J. Watson Research Center, enables users, user management, and the site's Computing Center to monitor and control usage of various computing resources. If a user's consumption of a particular resource exceeds a previously established limit, RESLIM takes actions designed to improve system performance and resource availability. Possible actions include degrading the user's priority, forcing the user off the system, or simply sending a warning message to the user and/or other VM users.

Volume 21, Number 1, 1982 *G321-0068*

Strategies for information requirements determination *by G. B. Davis, p. 4.* Correct and complete information requirements are key ingredients in planning organizational information systems and in implementing information systems applications. Yet, there has been relatively little research on information requirements determination, and there are relatively few practical, well-formulated procedures for obtaining complete, correct information requirements. Methods for obtaining and documenting information requirements are proposed, but they tend to be presented as general solutions rather than alternative methods for implementing a chosen strategy of requirements determination.

This paper identifies two major levels of requirements: the organizational information requirements reflected in a planned portfolio of applications and the detailed information requirements to be implemented in a specific application. The constraints on humans as information processors are described in order to explain why "asking" users for information requirements may not yield a complete, correct set. Various strategies for obtaining information requirements are explained. Examples are given of methods that fit each strategy. A contingency approach is then presented for selecting an information requirements determination strategy. The contingency approach is explained both for defining organizational information requirements and for defining specific, detailed requirements in the development of an application.

Business Systems Planning and Business Information Control Study: A comparison *by J. A. Zachman, p. 31.* Business Systems Planning (BSP) and Business Information Control Study (BICS) are two information system planning study methodologies that specifically employ enterprise analysis techniques in the course of their analyses. Underlying the BSP and BICS analyses are the data management problems that result from systems design approaches that optimize the management of technology at the expense of managing the data. In comparing BSP and BICS, five similarities and five differences are selected for discussion, and, finally, the strengths and weaknesses of each methodology are noted. The choice between using one or the other methodology is strongly influenced by the immediate intent of the study sponsor, tempered by the limiting factors currently surrounding the BICS methodology.

Supporting Business Systems Planning studies with the DB/DC Data Dictionary *by J. G. Sakamoto and F. W. Ball, p. 54.* Traditionally, Business Systems Planning (BSP) studies have been conducted using manual techniques. This paper describes one approach for computer assistance to such a study. The Extensibility facility of the IBM DB/DC Data Dictionary is shown to satisfy the requirements for the capturing of and subsequent reporting on BSP study data. The possibility of extending this approach to follow-on software development activities is discussed. General overviews of the Business

Systems Planning methodology and the IBM DB/DC Data Dictionary are also provided.

Towards an integrated development environment *by P. S. Newman, p. 81.* Problems of application-system costs, control, and effectiveness can best be addressed by highly consistent development and execution environments. This paper examines some relevant new approaches (systems description languages, new data models, application generators, and very-high-level languages), discusses the need for additional integration, and outlines a particular integration direction. This direction is intended to illustrate both the kind of consolidation needed and some of the problems involved.

Enterprise information analysis: Cost-benefit analysis and the data-managed system *by M. M. Parker, p. 108.* Enterprise information analysis studies have highlighted a gradual change in the data and information processing environment—a change in systems design and implementation from stand-alone, application-oriented systems, supporting primarily the operational and functional management levels, to data-base-oriented, data-managed systems, supporting the total organization. This shift has made many of the "traditional" financial analysis techniques used to justify a proposed system inadequate. Although a management study team that is developing an information systems proposal can choose from a variety of enterprise information needs, no such choice of associated (and generally accepted) disciplines or methodologies exists to support the financial justification of what has been proposed in the study team report. This paper explores the problems associated with moving from a "traditional" (data processing) financial justification of a system that is based largely on measurable costs and benefits to a financial justification of a system based largely on an assessment of intangible costs and benefits, technological change, and risk and uncertainty. A taxonomy is provided which can be used to supplement the value analysis found in the Business Systems Planning methodology. Extensive references are included as a guide to supplementary reading.

Volume 21, Number 2, 1982 *G321-0069*

Management considerations for an Information Center *by L. W. Hammond, p. 131.* This paper discusses what should be done in setting up an Information Center as part of an Information Systems group within a business organization. The Information Center is defined, including a user's viewpoint. Three key areas—the mission, organization and position, and staffing—are addressed. A procedure on how to initiate the center is presented. In general, the paper shows what an Information Center environment can be and how it might fit into a business organization.

How data flow can improve application development productivity *by W. P. Stevens, p. 162.* This paper presents the technique of data flow and how it can substantially improve application development productivity. Flows of data are the only connections needed between functional components of a computer program. Components which pass only data are so independent that they can easily be shared and reused. Such components can be developed independently, which substantially reduces the complexity of development and makes them much easier and faster to design, implement, test, and change. Building programs in this way can yield substantial increases in productivity over developing monolithic programs or even structures of called modules. The compatibility of data flow to natural human views of applications and other parts of data processing, such as distributed processing and high-performance architectures, is also presented. Recommendations are included.

SNA flow control: Architecture and implementation *by F. D. George and G. E. Young, p. 179.* To allow better network utilization, Systems Network Architecture (SNA), the IBM data communications architecture, includes flow control procedures to guard against data overrun to devices and to prevent network congestion. The measurement of "congestion" used by SNA to regulate traffic flow is performed by various SNA products. This paper describes the flow control protocols in SNA and the implementation of these protocols in the Network Control Program (ACF/NCP/VS Release 3).

Technique for assessing external design of software *by R. J. Pearsall, p. 211.* Discussed is a methodology of creating and using scenarios to assess completeness, correctness, consistency, and usability of the external design of computer software. Scenarios are paper tests of the specifications of software being designed. The approach is an outside-in, user-oriented evaluation of programs. The technique requires no machine time to perform the evaluation. As a result, defects are identified and changes are recommended early in the design phase of software development, at the time when defect removal costs are lowest.

The Document Interchange Architecture: A member of a family of architectures in the SNA environment *by T. Schick and R. F. Brockish, p. 220.* A wide variety of products for the office is now available, permitting increased automation of office procedures. To realize their full potential, these products must be able to exchange information and control requests with one another. A family of architectures has been defined to satisfy this need. This paper provides an overview of this family of architectures, including their relationship to one another. One member of this family, the Document Interchange Architecture, is described in some detail. An example illustrates use of the family of architectures in an office environment.

Volume 21, Number 3, 1982 *G321-0070*

JANUS: An interactive document formatter based on declarative tags *by D. D. Chamberlin, O. P. Bertrand, M. J. Goodfellow, J. C. King, D. R. Slutz, S. J. P. Todd, and B. W. Wade, p. 250.* This paper describes the architecture of an experimental document composition system named JANUS, which is intended to support authors of complex documents containing mixtures of text and images. The JANUS system is highly interactive, providing authors with immediate feedback and direct electronic control over page layouts, using a special two-display workstation. Authors communicate with the system by marking up their documents with high level descriptive "tags." A tag definition language is provided whereby new tags may be defined and the format of each tagged object may be controlled.

Office-by-Example: A business language that unifies data and word processing and electronic mail *by M. M. Zloof, p. 272.* The age of the nonprogrammer user of computing systems is at hand, bringing with it the special need of persons who are professionals in their own right to have

easy ways to use a computing system. Through the programming language discussed in this paper, executives and other office personnel can perform data and word processing and communications via terminals. This language, called Office-by-Example, provides rich and powerful access to the computing system computation, data base, communication, and display facilities. Discussed and illustrated by examples are a two-dimensional screen editor, triggers, and data bases, as well as word processing, electronic mail, customized menus, and application development.

The EPISTLE text-critiquing system *by G. E. Heidorn, K. Jensen, L. A. Miller, R. J. Byrd, and M. S. Chodorow, p. 305.* The experimental EPISTLE system is intended to provide "intelligent" functions for processing business correspondence and other texts in an office environment. This paper focuses on the initial objectives of the system: critiquing written material on points of grammar and style. The overall system is described, with some details of the implementation, the user interface, and the three levels of processing, especially the syntactic parsing of sentences with a computerized English grammar.

OPAS: An office procedure automation system *by V. Y. Lum, D. M. Choy, and N. C. Shu, p. 327.* This paper discusses an experimental system being developed to support office automation. The emphasis of the paper is on a technology that allows people to automate their office and business activities. Specifically, using forms as the interface, the authors propose a powerful data manipulation and restructuring facility that not only allows users to extract and manipulate data in the forms, but can be used to interface between new and existing applications as well.

Since business and office procedures are not discrete activities, but a structured sequence of activities, a means to define and execute procedures is required. Such a means is described in this paper along with its model and an example of its application.

A case study of office workstation use *by C. V. Bullen, J. L. Bennett, and E. D. Carlson, p. 351.* This paper describes the use of the Office Analysis Methodology to study a research office environment in order to determine requirements for an advanced office workstation. The research site environment is unique in providing an opportunity to observe a natural growth pattern in the use of advanced technology. Specific workstation requirements are identified and are being implemented. Interesting observations are reported in the following areas: categories of secretarial work, use of existing workstations, influence of a community of users, access to shared services, and effects on productivity and organizational behavior.

Volume 21, Number 4, 1982 *G321-0071*

The design rationale of the System/38 user interface *by J. H. Botterill, p. 384.* This paper is a discussion of the rationale behind the design of the software user interface of the System/38. It presents the design approaches used to produce a highly usable interactive system. The three primary system user interfaces are also presented, showing how the approaches were used in their design.

How a computer should talk to people *by M. Dean, p. 424.* This essay deals with a very down-to-earth topic: the things computer programs or systems say to people—in particular, computer messages for users. It is the product of the author's experience as a programmer and technical writer and editor. It puts together a lot of common-sense insights into the philosophy of creating good computer messages, how people think and feel around computers, how to analyze the situations in which people need a message, what to say in a message and how to say it, why imagination is invaluable for creating and evaluating messages, what technical questions must be answered in order to design and build a program or system that can talk effectively to people.

Analytic queuing model for CICS capacity planning *by M. Deitch, p. 454.* In recent years there has been a growing need to develop techniques and tools for computer installation capacity planning. This paper presents both a tool, in the form of an analytic queuing model, and a methodology for performance analysis and capacity planning. Although general in its approach, the model was developed specifically for the CICS/VS environment.

Modeling distributed processing across multiple CICS/VS sites *by R. D. Acker and P. H. Seaman, p. 471.* Modeling is a useful method to aid a planner in designing the interconnection of a number of systems for distributed data processing. In the implementation in this paper, a computer-based model for sites using CICS/VS is discussed. The model permits the system definition to be adjusted, taking into account such aspects as the number of sites. their interconnections, and workloads, so that a satisfactory configuration can be obtained.

IMS/VS: An evolving system *by J. P. Strickland, P. P. Uhrowczik, and V. L. Watts, p. 490.* The information Management System, IMS, began in the mid-1960s as a batch-only data base system that was known then as Data Language/I (DL/I). IMS was introduced in 1969 as IMS/360, a program product for the System/360. As the System/360 evolved into System/370, including support for virtual storage, the operating system evolved into OS/VSI, OS/VS2, and then MVS. At the same time, IMS evolved to become IMS/VS. The Information Management System has continued to be adapted to new requirements, especially those of interactive, on-line operations that require data communications. Recent advances in the following categories of IMS/VS functions are discussed in this paper: Fast Path, Data Sharing, System Logging, Data Base Recovery Control, on-line changes in system environment, Intersystem Communications, MVS Common Services Area usage, and architectural restructuring.

Volume 22, Number 1/2, 1983 *G321-0072*

A perspective on communications and computing *by A. L. Scherr, p. 5.* Presented is an essay on the dynamics of the relationships between communications and computing. Movement of computer applications from the back office to the front office, from batch to on-line data processing, is illustrated and conclusions are drawn regarding communications protocols, network management, application and data base design, and system generality. The influence and requirements of several new technologies are presented, including those of microprocessors and teleconferencing.

X.25 and related recommendations in IBM products *by G. A. Deaton, Jr. and R. O. Hippert, Jr., p. 11.* This paper describes IBM's use of Recommendation X.25 and re-

lated recommendations of the International Telegraph and Telephone Consultative Committee. After reviewing the development history of X.25 and some of the motivations for using it, the paper gives an overview of packet-switched data networks. The reader is then given a brief technical description of Recommendation X.25 and some other recommendations used in conjunction with X.25. The architectural relationships between X.25 and IBM's Systems Network Architecture (SNA) are described for packet-switched X.25 connections between SNA and non-SNA nodes. Specific elements of Recommendation X.25 used in SNA nodes are defined. After several IBM products that support X.25 and some of the related recommendations are described, IBM's equipment for testing the X.25 interface is discussed.

Teletex–A worldwide link among office systems for electronic document exchange *by D. J. Moore, p. 30.* Teletex is a new international telecommunication service that provides direct electronic document exchange between such office text machines as electronic typewriters and word processors that are equipped with transmitting and receiving storages. Teletex is an international standard aimed at integrating office products and worldwide telecommunication. It represents a major step in the development of the office of the future. This paper traces the development of Teletex, describes its characteristics, and looks at how this service may be extended in the future.

A token-ring network for local data communications *by R. C. Dixon, N. C. Strole, and J. D. Markov, p. 47.* Technical innovations such as large-scale integrated circuit technology and distributed operating systems have respectively reduced the cost of computing and provided a basis for large networks within the confines of a single building or cluster of buildings in close proximity to one another. Local area networks can provide a systematic approach for interconnecting personal workstations, control units, and central processing units, thereby providing a means for these machines to pass information from one to the other. This paper describes a local area network based on the fundamental concepts of a token-ring. Two main ideas are presented. The first idea concerns the physical topology of the wiring network and its star-ring organization. Next, the logical data flows are overlaid on the physical network to provide control procedures for exchanging data through the network. The resulting system has unique features that produce a local area network with good performance and reliability characteristics.

Reflections on VM/Pass-Through: A facility for interactive networking *by N. Mendelsohn, M. H. Linehan, and W. J. Anzick, p. 63.* VM/Pass-Through, an interactive networking facility, has gained widespread acceptance within IBM and with IBM customers. Pass-Through allows a single terminal access to many different computers, including those at distant locations. In building Pass-Through, and in observing its growing use, we have had an opportunity to study the practical implications of this facility and of our approach to its design.

This paper is divided into two parts. The first, an introduction to Pass-Through networking, describes features of the system, supported configurations, and use of Pass-Through within the IBM Corporation. A brief history of Pass-Through's development is also provided. In the second part of the paper, Pass-Through is used to motivate a technical discussion of interactive network technology and virtual machine subsystems. Topics covered include appropriate use of the virtual machine environment, choice of routing strategy, and performance considerations. Although the introductory portions of the paper presume no prior knowledge of computer network or operating system technology, the subsequent technical discussions do depend on a basic understanding of these areas.

A Satellite Communications Controller *by J. W. Fennel, Jr. and B. D. Gobioff, p. 81.* A satellite communications controller, which is a component of satellite earth stations, establishes a time division multiple access structure that provides flexibility and efficiency in the use of satellite transmission capacity. The controller blends digital, computer, and communications technology to establish precise timings required for system synchronization. It also performs signal conversion and switching necessary to transmit voice and business machine traffic over the satellite. This paper discusses a particular satellite communications system and the architecture and implementation of the satellite communications controller.

Series/1-based videoconferencing system *by D. Anastassiou, M. K. Brown, H. C. Jones, J. L. Mitchell, W. B. Pennebaker, and K. S. Pennington, p. 97.* Discussed is a new videoconferencing system that has been developed and deployed at several IBM locations. This system transmits high-quality monochrome, freeze-frame images over dial-up telephone lines between two (or three) dedicated videoconferencing rooms. There are two main system components. An IBM Series/1 provides control, communication, data compression, and storage, and Grinnell GMR-270 image processing display system implements image acquisition, processing, and video buffering functions. Conference participants may choose either a basically black and white rendering of an image for fast transmission or a continuous-tone rendering with a longer transmission time. Details are given regarding the system configuration, function, and operation.

NIL: A high-level language for distributed systems programming *by F. N. Parr and R. E. Strom, p. 111.* Network Implementation Language (NIL) is a high-level programming language currently being used for the implementation of prototype communication systems. NIL is designed for writing executable architecture which can be compiled into efficient code for the different machines and run-time environments of a family of communication products. NIL's distinctive features include (1) high-level primitive type families supporting constructs needed for concurrent systems, (2) facilities for decomposition of a system into modules which can be dynamically installed and interconnected, (3) compile-time *typestate checking*—a mechanism for enhancing language security without incurring large execution-time overhead.

Communications Network Management enhancements for SNA networks: An overview *by T. P. Sullivan, p. 129.* Hierarchical growth and diversification of communications networks have imposed new requirements on Communications Network Management. This essay presents the evolution of support included in IBM's Systems Network Architecture to meet these needs. Four main provisions of network management are discussed: (1) the collection and presentation of downstream network data on behalf of network resources by a Threshold Analysis and Remote Access program; (2)

centralized network operator terminal access to local and remote systems by a Terminal Access Facility; (3) unsolicited alerting of the network operator by the presentation of data from network resources by a Network Problem Determination Application; and (4) the collection and presentation of data for the logical network by a Network Logical Data Manager.

An application of network management at a large computing service *by R. D. Garrigues, p. 143.* Productive operation of a large computer network serving multiple user facilities in conjuction with several host sites requires the use of network management. As treated here, network management includes an array of such methods as communication systems management, problem management, change management, and inventory management. In this paper the use of some of the management techniques is described as they apply in an education support facility network. The setup of the facility is first described; then some of the techniques for managing its operation are presented.

Volume 22, Number 3, 1983 *G321-0073*

Abstract design and program translator: New tools for software design *by J. L. Archibald, B. M. Leavenworth, and L. R. Power, p. 170.* Abstract Design And Program Translator (ADAPT) is an integrated set of tools and approaches for the design and development of software systems. Together they include a module specification language and a system design language for specifying module interfaces and interconnections. This paper explains some of their major features and illustrates their use in the design of some examples—a set of reusable software components and a generalized editor system. Benefits of the ADAPT approach are discussed, emphasizing executable design and modifiability.

The system architecture of EAS-E: An integrated programming and data base language *by D. P. Pazel, A. Malhotra, and H. M. Markowitz, p. 188.* EAS-E is an application development system based on an entity-attribute-set view of system description. It consists of a procedural language for manipulating data base and main storage entities, and direct (nonprocedural) facilities for interrogating and updating data base entities. the EAS-E software itself was implemented with the entity-attribute-set view. This paper reviews some of the EAS-E features and considers some of its implementation details. This paper is both an introduction to the EAS-E software architecture and an example of the usefulness of the entity-attribute-set view.

A simple architecture for consistent application program design *by G. R. Rogers, p. 199.* This paper addresses the architectural design aspects of general business computer application programs written in high-level procedural programming languages. It puts forth design concepts for easily built, maintainable programs and describes a unique approach to program decomposition.

The Project Automated Librarian *by J. M. Prager, p. 214.* The Project Automated Librarian (PAL) is a tool that has been created to manage the logistical problems inherent in a medium-sized software development project. The main goals of PAL are to eliminate the problems of simultaneous updates to software modules, while allowing programmers access to the latest possible versions of the software. PAL also seeks to prevent the software from getting into an inconsistent state that could prevent users from proceeding with software development because of someone else's errors. PAL is a general-purpose tool, in the sense that it does not care what language or languages the system is being written in. It makes backups, keeps version information, and maintains documentation of changes.

Automatic generation of random self-checking test cases *by D. L. Bird and C. U. Munoz, p. 229.* A technique of automatically generating random software test cases is described. The nature of such test cases ensures that they will execute to completion, and their execution is predicted at the time of generation. Wherever possible the test cases are self-checking. At run-time their execution is compared with the predicted execution. Also described are implementations of the technique that have been used to test various IBM programs–PL/I language processors, sort/merge programs, and Graphical Data Display Manager alphanumeric and graphics support.

Full-screen testing of interactive applications *by M. E. Maurer, p. 246.* This paper describes the dialog test functions of the Interactive System Productivity Facility/Program Development Facility program product, with emphasis on the full-screen design that makes it unique. Perspective is provided by a brief summary of the test facilities available in the predecessor System Productivity Facility program product (SPF) and the requirements that led to their enhancement.

Software reliability analysis *by P. N. Misra, p. 262.* Methods proposed for software reliability prediction are reviewed. A case study is then presented of the analysis of failure data from a Space Shuttle software project to predict the number of failures likely during a mission, and the subsequent verification of these predictions.

Design and use of a program execution analyzer *by L. R. Power, p. 271.* Execution analyzers are used to improve the performance of programs, operating systems, and hardware systems. This paper presents a general overview of these tools, especially those designed for use by application programmers. The design tradeoffs of a wide variety of execution analyzers are examined. In addition, the design and use of a new execution analyzer are presented; its purpose is to assist in the optimization of highly modular PL/I programs.

Volume 22, Number 4, 1983 *G321-0074*

Advanced program-to-program communication in SNA *by J. P. Gray, P. J. Hansen, P. Homan, M. A. Lerner, and M. Pozefsky, p. 298.* Systems Network Architecture (SNA) defines the behavior of networks of heterogeneous, loosely coupled processors. This paper describes the development of program-to-program communication services in SNA and introduces Advanced Program-to-Program Communication (APPC), the culmination of this development. It also discusses the use of APPC in the construction of distributed services and shows that SNA with APPC and other SNA services can be thought of as a distributed operating system.

SNA Distribution Services *by B. C. Housel and C. J. Scopinich, p. 319.* This paper describes the IBM SNA Distribution Services (SNADS). Heretofore, SNA has focused on synchronous data distribution. Along with the advent of office systems and other distributed applications has come the requirement to provide a common architecture for interchanging data asynchronously

among diverse systems and products. SNA Distribution Services provides a general asynchronous (delayed delivery) data distribution facility for SNA applications. The initial implementations are for office systems applications. Discussed are objectives for an asynchronous data distribution service, key architectural concepts, the relationship between SNADS and SNA synchronous communications architecture, and the interface between the distribution service and application transaction programs.

Interconnecting SNA networks *by J. H. Benjamin, M. L. Hess, R. A. Weingarten, and W. R. Wheeler, p. 344.* Systems Network Architecture (SNA) allows terminals and application programs to communicate with one another using SNA entities called logical units. Until now, these logical units have had to be in the same network to communicate. This paper describes recently introduced SNA network interconnection functions that allow logical units in independent SNA networks to communicate with one another. Each network is configured, defined and managed separately. By using one or more facilities called gateways, networks can remain independent while their logical units initiate, use, and terminate internetwork sessions, without any changes to themselves. A communications user need not be aware that a session partner is in a separate network.

An experimental address space isolation technique for SNA networks *by K. D. Ryder, p. 367.* The integration of computer networks has led to increasingly large and complex configurations. This integration has resulted in concern for the availability of resources for the merged networks. An experimental technique called TRAP has been developed as a way to minimize the constraints on such networks. It allows address space isolation between interconnected networks. This paper discusses the technique and its fundamental process of network address translation.

Logical problem determination for SNA networks *by R. A. Weingarten and E. E. Iacobucci, p. 387.* Problem determination on a Systems Network Architecture network has dealt mostly with error detection on physical network components. Adequate logical error-detection mechanisms associated with the logical network (software-related) errors have been only recently provided with the announcement of a new on-line interactive package called the Network Logical Data Manager (NLDM). This paper discusses the physical and logical network environments, logical network problems, and functions provided by the two releases of NLDM for logical problem determination.

Performance and availability measurement of the IBM Information Network *by R. M. Bailey and R. C. Soucy, p. 404.* A key requirement of a network service is the management of specified service levels as perceived by the end user. A capability for measuring and reporting end user response time and availability is essential. This paper describes measurement techniques to track these key service-level attributes in the IBM Information Network (IBM/IN). These techniques apply to most complex SNA networks.

SNA routing: Past, present, and possible future *by J. M. Jaffe, F. H. Moss, and R. A. Weingarten, p. 417.* This paper reviews the evolution of routing mechanisms in IBM's Systems Network Architecture (SNA) since its inception in 1974 to the present. Routing mechanisms are related to changes in the application and communications environment. Also discussed are possible evolutionary paths that may be taken in the future to address the problems of large heterogeneous networks.

Defining routing tables for SNA networks *by K. Maruyama, p. 435.* This paper addresses three basic problems associated with the definition process for the routing tables of IBM's Systems Network Architecture (SNA). The paper then introduces a program called the Routing Table Generator (RTG) and describes how these problems were solved with RTG. Also discussed are some approaches on how to use RTG in managing routing tables for growing networks.

Windows in the sky–Flow control in SNA networks with satellite links *by G. A. Grover and K. Bharath-Kumar, p. 451.* Geosynchronous communications satellites provide a unique means of high-speed computer-to-computer transmission of large volumes of data over long distances. The physical distances involved in transmitting to and from the satellites cause relatively long propagation delays for messages. In order that the high bandwidth be used effectively, large quantities of data have to be transmitted before pausing for an acknowledgment. This condition creates a potential for some new and unique types of traffic jams. This paper discusses these situations in the context of Systems Network Architecture (SNA) networks. In particular, the issues related to SNA's flow control and traffic management facilities in the presence of satellite links are discussed, along with potential solutions to ensure efficient network operation.

Volume 23, Number 1, 1984 *G321-0075*

Architecture prototyping in the software engineering environment *by W. E. Beregi, p. 4.* This technical essay presents a perspective on the evolution and problems of the software development craft and how software engineering techniques show promise to solve these problems. It introduces architecture prototyping as a program development technique for improving software quality. Experience with large software systems shows that over half of the defects found after product release are traceable to errors in early product design. Furthermore, more than half the software life-cycle costs involve detecting and correcting design flaws. In this paper, we explore a disciplined approach to software development based on the use of formal specification techniques to express software requirements and system design. As a consequence, we can use techniques like rapid prototyping, static design analysis, design simulation, and dynamic behavior analysis to validate system design concepts prior to element design and implementation. We explore how these techniques might be organized in a software architecture prototyping facility that would be similar to the Computer-Aided Design and Manufacturing (CADAM) tools used in other engineering disciplines. We also examine the process by which software engineers might use these facilities to create more reliable systems.

Factors affecting programmer productivity during application development *by A. J. Thadhani, p. 19.* The effects of good computer services on programmer and project productivity during application program development are examined. Programmer's terminal activity and the nature of terminal work are analyzed. The discussion

includes the effects of short response times, programmers' skills, and program complexity on productivity.

A comparative study of system response time on program developer productivity *by G. N. Lambert, p. 36.* Skilled programmer time and computer time and resources are valuable. Earlier studies had shown that added computer resources can decrease system response time and increase programmer productivity significantly. A controlled study has been made to determine whether that finding is true for the particular conditions in another program development organization. That study is reported here. Programmer productivity increased sixty-two percent with subsecond system response time. A new finding is that individual group project offices lead to greater efficiency than large open rooms.

Analysis of free-storage algorithms–revisited *by G. Bozman, W. Buco, T. P. Daly, and W. H. Tetzlaff, p. 44.* Most research in free-storage management has centered around strategies that search a linked list and strategies that partition storage into predetermined sizes. Such algorithms are analyzed in terms of CPU efficiency and storage efficiency. The subject of this study is the free-storage management in the Virtual Machine/System Product (VM/SP) system control program. As a part of this study, simulations were done of established, and proposed, dynamic storage algorithms for the VM/SP operating system. Empirical evidence is given that simplifying statistical assumptions about the distribution of interarrival times and holding times has high predictive ability. Algorithms such as first-fit, modified first-fit, and best-fit are found to be CPU-inefficient. Buddy systems are found to be very fast but suffer from a high degree of internal fragmentation. A form of extended subpooling is shown to be as fast as buddy systems with improved storage efficiency. This algorithm was implemented for VM/SP, and then measured. Results for this algorithm are given for several production VM/SP systems.

Speech filing–An office system for principals *by J. D. Gould and S. J. Boies, p. 65.* Business people spend most of their time communicating, or attempting to communicate, with others. We briefly describe our ideas about these communication activities and their resulting problems, and then discuss an experimental tool we developed to help business people solve some of their communication problems. This tool, called the Speech Filing System, allows users to send messages to anybody in the world. The system offers powerful editing, filing, retrieval, and message distribution and control functions, using pushbutton telephones as the terminals.

Playback: A method for evaluating the usability of software and its documentation *by A. S. Neal and R. M. Simons, p. 82.* Human factors evaluations of software products and accompanying user publications must be conducted so that developers can be certain that the target user population can learn to use the product with a minimum of difficulty and be able to perform the intended tasks efficiently. A methodology is described for obtaining objective measures of product usability by collecting performance data on the user interface without affecting the user or the system being evaluated. The log of stored activity is later played back through the host system for analysis.

An overview of three relational data base products *by S. Kahn, p. 100.* This issue of the *IBM Systems Journal* focuses on aspects of three recently announced IBM relational data base products. They are IBM Database 2, Query Management Facility, and Data Extract. This essay illustrates the requirements that these products were designed to address and gives a brief overview of their content and history. Its objective is to provide an introduction to the more specific and detailed papers that follow.

IBM Database 2 overview *by D. J. Haderle and R. D. Jackson, p. 112.* IBM Database 2 (DB2) is a data base management system that supports the relational model of data. This paper presents the major features of DB2 and discusses its architecture and the relationship of DB2 with the host operating system. These principles are illustrated by an example.

The Query Management Facility *by J. J. Sordi, p. 126.* Data from a relational data base can be displayed in reports, changed, and otherwise controlled using a program called Query Management Facility (QMF). An overview of this program is presented and is followed by a discussion comparing equivalent forms of various queries expressed in two distinctly different languages. Both languages are designed for use with relational data and are supported by QMF.

TSO Attach: A multipurpose communication channel to IBM Database 2 *by K. R. Hammond and M. R. Zimowski, p. 151.* TSO Attach provides IBM Database 2 capabilities in a productive work environment that appears as a natural extension of the Time Sharing Option (TSO) and the Interactive System Productivity Facility (ISPF). It was designed and built with careful consideration for the varied and complex user group for which it was intended. Ease of use and ease of development and maintenance were among the significant factors in the design. These factors and others are addressed in this paper, which discusses the basic design decisions made in building the TSO Attachment Facility.

IBM Database 2 in an Information Management System environment *by J. R. Dash and R. N. Ojala, p. 165.* Over the years, the IBM Information Management System (IMS/VS) has been developed to meet expanding under needs. During that time, a parallel development has taken place. The relational data model grew from Codd's original theory to a practical data base prototype. Now a new data base management system, IBM Database 2 (DB2), has been built on the relational model. This paper discusses the implementation and design considerations for the integration of IMS and DB2 from the user's viewpoint. It also presents the attachment facilities from a design perspective.

Data recovery in IBM Database 2 *by R. A. Crus, p. 178.* This paper presents the various forms of data recovery provided by IBM Database 2 (DB2). It describes the DB2 recovery log, introduces the notion of a unit of recovery, and discusses the two-phase commit protocol used by DB2. Furthermore, it describes what type of information is logged, the DB2 checkpoint process, what a compensation log record is, and how DB2 handles undo/redo processing, media recovery, restart after abnormal system termination, and data unavailability.

IBM Database 2 performance: Design, implementation, and tuning *by J. M. Cheng, C. R. Loosley, A. Shibamiya, and P. S. Worthington, p. 189.* The larger and more complex a relational data base, the more efficient the data base management system must be to maintain an acceptable level of performance. The design and implementation of IBM Database 2 (DB2) have been aimed toward this objective. Techniques for achieving this key objective in DB2 are the subject of this paper. Presented are performance-related strategies in query processing and performance-related design tradeoffs. Data base and application design options and their resolution for optimum performance are also discussed. Also presented are techniques to maintain performance by application monitoring and tuning and DB2 system tuning.

Managing IBM Database 2 buffers to maximize performance *by J. Z. Teng and R. A. Gumaer, p. 211.* The relational data base system, IBM Database 2 (DB2), has a component that manages data buffering. This paper describes the design considerations of the Buffer Manager and the tradeoffs involved in managing the allocation of DB2 buffers to maximize performance.

Volume 23, Number 3, 1984 *G321-0077*

Ease of use: A system design challenge *by L. M. Branscomb and J. C. Thomas, p. 224.* While it is becoming increasingly obvious that the fundamental architecture of a system has a profound influence on the quality of its human factors, the vast majority of human factors studies concern the surface of hardware (keyboards, screens) or the very surface of the software (command names, menu formats). In this paper, we discuss human factors and system architecture. We offer best-guess guidelines for what a system should be like and how it should be developed. In addition, we suggest ways in which advances in research and education could result in systems with better human factors. This paper is based on an address by L. M. Branscomb and a publication by the authors in the Proceedings of the IFIP 9th World Computer Congress, Paris, France, September 19–23, 1983.

Directions in cooperative processing between workstations and hosts *by B. C. Goldstein, A. R. Heller, F. H. Moss, and I. Wladawsky-Berger, p. 236.* Advancements in technology have provided us with the availability of high-performance processors from the high end of computing to the personal computer. In addition, technology growth has enabled us to envision sixteen megabytes of real storage for a personal computer.

As a result, we have witnessed not only a tremendous growth at the high end of the computing spectrum, but also the development of sophisticated personal computers (e.g., the IBM PC XT/370) with real storage capacities approaching those of high-end computers of a decade ago.

This growth at both ends of the computing spectrum has given us a choice. We can either allow a clean separation to grow between personal computer and host or provide a means by which they cooperate in providing quality service to the user without the complexity normally associated with high-end systems. This paper explores what such a cooperation could mean.

System/370 capability in a desktop computer *by F. T. Kozuh, D. L. Livingston, and T. C. Spillman, p. 245.* A desktop computer with System/370 capability was produced by enhancing the IBM Personal Computer XT with additional hardware and developing software that provides a compatible interface. The computer, the IBM Personal Computer XT/370, and this software allow users to run most System/370 Conversational Monitor System application programs unaltered in a desktop environment. The evolution of the development and details of the function of the hardware and software are described.

A tight coupling of workstations *by D. M. Chess, p. 255.* This paper addresses the problem of situations in which people at physically distant locations must have access to essentially the same computing environment at the same time. That is, each user must be able to provide input to whatever application or system is active, and must be provided with all relevant output. Common examples of this situation are demonstrations, presentations, education, and troubleshooting.

A prototype system has been developed to study ways of solving this problem in the microcomputer workstation environment. The prototype allows users at two IBM Personal Computers to share access to the computing environment through the keyboard and the display screen by tightly coupling the computers.

Architecture implications in the design of microprocessors *by R. E. Matick and D. T. Ling, p. 264.* This paper examines how architecture, the definition of the instruction set and other facilities that are available to the user, can influence the implementation of a very large scale integration (VLSI) microsystem. The instruction set affects the system implementation in a number of direct ways. The instruction formats determine the complexity of instruction decoding. The addressing modes available determine not only the hardware needed (multiported register files or three-operand adders), but also the complexity of the overall machine pipeline as greater variability is introduced in the time it takes to obtain an operand. Naturally, the actual operations specified by the instructions determine the hardware needed by the execution unit. In a less direct way, the architecture also determines the memory bandwidth required. A few key parameters are introduced that characterize the architecture and can be simply obtained from a typical workload. These parameters are used to analyze the memory bandwidth required and indicate whether the system is CPU- or memory-limited at a given design point. The implications of caches and virtual memories are also briefly considered.

Use of images in commercial and office systems *by P. J. Somerville, p. 281.* This paper examines some of the simpler processing techniques that may usefully be performed on bi-level (two-tone) images by a competent commercial applications programmer if a few basic (albeit complex internally) tools are provided. Such processes include storage, indexing, changing resolution, rotating, trimming, and then display and printing. These processes can provide the facilities for an enterprise to incorporate images and image data into its office systems and into its main line-of-business data processing applications.

Security considerations for personal computers *by W. H. Murray, p. 297.* The wide use of personal computers and general access to telecommunications links have intensified the need for computer security. Security practices

as discussed in this paper relate to protecting an organization's personal computers as physical property, protecting the organization's data and applications, and protecting the organization itself. These matters are discussed from the point of view of protection from the improper use of personal computers.

Volume 23, Number 4, 1984 *G321-0078*

An overview of computer security *by R. C. Summers, p. 309.* Presented is an overview of computer security, including concepts, techniques, and measures relating to the protection of computing systems and the information they maintain against deliberate or accidental threats. Motivations for security measures are discussed. Security strategies are considered. Actions and events that threaten security are described, along with technical problems that can prevent the computer from adequately dealing with threats. Security models are surveyed. Specific technical and administrative measures for promoting security are described. Among the technical measures discussed are design of secure systems, hardware and operating systems, identification of users, encryption, and access control packages. Administrative measures include personnel, physical security of the computing system, and auditing. Also presented is the establishment of a security program. Reviewed are special problems and their solutions, including communications and networks, data base management systems, and statistical data bases. This paper is based on a paper by the author published in *The Handbook of Computers and Computing*, edited by Arthur H. Seidman and Ivan Flores, Van Nostrand Reinhold Company, Inc., New York (1984).

The design of the REXX language *by M. F. Cowlishaw, p. 326.* One way of classifying computer languages is by two classes: languages needing skilled programmers, and personal languages used by an expanding population of general users. REstructured eXtended eXecutor (REXX) is a flexible personal language designed with particular attention to feedback from its users. It has proved to be effective and easy to use, yet it is sufficiently general and powerful to fulfill the needs of many demanding professional applications. REXX is system and hardware independent, so that it has been possible to integrate it experimentally into several operating systems. Here REXX is used for such purposes as command and macro programming, prototyping, education, and personal programming. This paper introduces REXX and describes the basic design principles that were followed in developing it.

An application analyzer *by R. Ambrosetti, T. A. Ciriani, and R. Pennacchi, p. 336.* An interactive tool, aimed at supporting the application user/analyst in specifying and analyzing a business area, is presented. The features of the tool, named the Application Analyzer/Experimental, are described both in their theoretical foundations and their actual implementation. A brief description of the architecture of the tool and its internal structure is given. A review of the main concepts of the application development area is also included. The follow-on of the prototype described here is the program offering known as System A.

Performance issues in local-area networks *by W. Bux, p. 351.* This paper discusses several important performance problems in the design of local-area networks. The questions discussed relate to various aspects of architecture, design, and implementation: (1) the delay-throughput characteristics of the medium access protocols, (2) the performance of local-area networks on which a file server provides file storage and retrieval services to intelligent workstations, and (3) timing problems in local-area network adapters. Since the paper does not primarily address the performance analyst, it is descriptive in nature; analytic details are omitted in favor of a more intuitive explanation of the relevant effects.

VM/370, Attached Processor, and multiprocessor performance study *by W. H. Tetzlaff and W. M. Buco, p. 375.* This paper discusses performance studies of Attached Processors, multiprocessors, and VM/370. A methodology for evaluating performance is discussed. Performance improvements are explained and evaluated. These studies played a role in a new option to the VM/System Product control program that is called the High Performance Option (HPO).

Volume 24, Number 1, 1985 *G321-0079*

Standardized graphics on the IBM Personal Computer *by T. B. Clarkson III, p. 3.* Although acknowledged to be an effective means of communicating information, graphics has not progressed more rapidly in the burgeoning use of personal computers due to the lack of standards for both writing and running graphics applications. A graphics standard—the Virtual Device Interface (VDI)—has been proposed for national use and is in the process of being adopted. An implementation of the VDI is currently available for the IBM Personal Computer. This paper briefly traces the history of graphics as used with personal computers, explores the difficulties that standardization efforts have met, explains the VDI model, and shows how this model operates in the IBM Personal Computer environment to make graphics a natural extension of the operating system.

A professional graphics controller *by K. A. Duke and W. A. Wall, p. 14.* The IBM Professional Graphics Controller and Display were developed to meet the needs of engineers and scientists for an improved graphics capability in the Personal Computer environment. These units provide graphics systems with improved function, resolution, and color range, and at the same time they allow existing productivity software to be executed in an emulation mode. This paper describes the function and discusses the design of the Professional Graphics Controller.

Expanded personal computing power and capability *by P. A. Korn, J. P. McAdaragh, and C. L. Tondo, p. 26.* Discussed is the XENIX™ Operating System for the IBM Personal Computer AT. The operating system incorporates capabilities of a mainframe operating system—multiusage, multitasking, file management and security, program compilation, and networking. The XENIX shell structure is introduced. Pipes and pipelining are presented. The XENIX file structure is explained and illustrated with examples. Software development and text formatting are treated in detail. The ability to compile C program code developed under XENIX and run it on the IBM Personal Computer Disk Operating System is explained.

The C programming language and a C compiler *by R. R. Ryan and H. Spiller, p. 37.* In the last few years, the C programming language has become one of the most widely used languages for applications and systems software on microcomputer systems. This paper describes the C language and its history. It then presents a specific implementation of C, the Microsoft C Compiler, which runs on the IBM Personal Computer.

Design considerations for IBM Personal Computer Professional FORTRAN, an optimizing compiler *by M. L. Roberts and P. D. Griffiths, p. 49.* An optimizing FORTRAN compiler with power to handle large applications at execution speeds comparable to those of large computers has been implemented on the IBM Personal Computer. This implementation is described with emphasis on the design decisions that were considered in the development of the compiler.

An APL system for the IBM Personal Computer *by M. L. Tavera, M. Alfonseca, and J. Rojas, p. 61.* This paper discusses the design and building of an APL interpreter for the IBM Personal Computer. Discussed is the writing of the interpreter itself, which required the use of an intermediate language designed by the authors. This machine-independent language also made possible the development of APL interpreters for two other systems—System/370 and Series/1. The particularizing of the interpreter required a compiler, which in the case of the Personal Computer produced Intel 8088 and 8087 assembly language code. The matching of the APL interpreter to the operating system (DOS) required an APL supervisor, which is also discussed in this paper. The provision of the APL character set presented problems, the solutions of which are also presented. Other topics discussed are the display, the keyboard, and the session manager.

Volume 24, Number 2, 1985 *G321-0080*

The IBM large-systems software development process: Objectives and direction *by W. S. Humphrey, p. 76.* This paper introduces a special issue of the *IBM Systems Journal* on the IBM large-systems software development process. The issue provides an overview of the subject and a summary of the key principles of the IBM software quality and productivity efforts in large-scale systems programming. The major topics addressed in this issue are the software development process, software development tools and methodologies, quality and productivity measurements, and programmer education.

A programming process architecture *by R. A. Radice, N. K. Roth, A. C. O'Hara, Jr., and W. A. Ciarfella, p. 79.* The Programming Process Architecture is a framework describing required activities for an operational process that can be used to develop system or application software. The architecture includes process management tasks, mechanisms for analysis and development of the process, and product quality reviews during the various stages of the development cycle. It requires explicit entry criteria, validation, and exit criteria for each task in the process, which combined form the "essence" of the architecture. The architecture describes requirements for a process needing no new invention, but rather using the best proven methodologies, techniques, and tools available today. This paper describes the Programming Process Architecture and its use, emphasizing the reasons for its development.

A programming process study *by R. A. Radice, J. T. Harding, P. E. Munnis, and R. W. Phillips, p. 91.* A programming Site Study group was convened to look at the work of eight large-system programming development locations within IBM and to evaluate them according to a set of process stages. Eleven attributes were applied to each process stage. The process of the Site Studies is directly transferable to software evaluations on any project in the software industy, and it is believed that the studies are the first step necessary in the evolution of a consistently repeatable and dynamically controllable process of improvement within the industry. The phases of these studies and implementation of the studies are described.

Automating the software development process *by G. F. Hoffnagle and W. E. Beregi, p. 102.* Demand for reliable software systems is stressing software production capability, and automation is seen as a practical approach to increasing productivity and quality. Discussed in this paper are an approach and an architecture for automating the software development process. The concepts are developed from the viewpoint of the needs of the software development process, rather than that of established tools or technology. We discuss why automation of software development must be accomplished by evolutionary means. We define the architecture of a software engineering support facility to support long-term process experimentation, evolution, and automation. Such a facility would provide flexibility, tool portability, tool and process integration, and process automation for a wide range of methodologies and tools. We present the architectural concepts for such a facility and examine ways in which it can be used to foster software automation.

Quality emphasis at IBM's Software Engineering Institute *by M. B. Carpenter and H. K. Hallman, p. 121.* Improvements in quality and productivity in the development of programs can be obtained by instructing the programming development groups in the use of modern software engineering methodology. To provide this instruction for its employees, IBM has established a Software Engineering Institute. Currently training in the methodology is being offered through an education program of the Institute known as the Software Engineering Workshop. This paper describes the role of the Institute, its background and offerings, and some results obtained.

PDM: A requirements methodology for software system enhancements *by R. G. Mays, L. S. Orzech, W. A. Ciarfella, and R. W. Phillips, p. 134.* Traditional requirements processes often do not address the many problems encountered in the development of software products. Conventional processes begin with the structural definition of the proposed system, under the assumption that the raw requirements are understood. How this understanding is developed is not formally addressed. The IBM software development process requires a methodology to develop the rationale of the requirement, both in terms of its underlying problem and its business justification, prior to the development of the functional specification. Conventional requirements processes address a single software application intended for use by a uniform set of end users. The resulting system is usually a one-time replacement of some existing system. Many IBM software products, however, address requirements received from a large, diverse set

of customers who use the products in a wide array of computing environments. Product releases are typically developed as incremental enhancements to an existing base product. This paper describes the Planning and Design Methodology (PDM), a requirements planning process that supports the collection, analysis, documentation, and tracking of software requirements. The process includes requirements collection, definition of the underlying problems, development of an external functional description that addresses the problems, and development of system and product designs from the external functional descriptions. PDM has been applied in three development areas with positive results.

A process-integrated approach to defect prevention *by C. L. Jones, p. 150.* Recent efforts to improve quality in software have concentrated on defect detection. This paper presents a programming process methodology for using causal analysis and feedback as a means for achieving quality improvements and ultimately defect prevention. The methodology emphasizes effective utilization of all error data to prevent the recurrence of defects.

Programming process productivity measurement system for System/370 *by M. J. Flaherty, p. 168.* Discussed in this paper are the underlying principles of a programmer productivity measuring system. The key measures (or metrics) are people and lines of code. Definitions of these metrics are refined and qualified, according to the conditions under which they are used. Presented also is a data base design for retaining and retrieving these metrics under a wide variety of applications and other circumstances. Depending on definitions, applications, and other circumstances, productivity measurements may differ widely. On the other hand, after suitable productivity metrics have been defined, consistency of application of the same metrics yields comparable results from project to project.

Volume 24, Number 3/4, 1985 *G321-0081*

Worldwide systems engineering *by T. G. Peck, p. 182.* IBM systems engineering celebrates its 25th anniversary in 1985. This paper provides a perspective of the part systems engineering has played in the success of IBM in the information processing business during that 25-year period. The history of systems engineering is briefly reviewed, and the similarities and differences in worldwide systems engineer functions are examined. The relationships among marketing, systems engineering, and customers are discussed. Also discussed are career paths for systems engineers. Expectations and challenges for systems engineering in the future are explored.

HONE: The IBM marketing support system *by W. Boos, p. 189.* The storage, retrieval, and dissemination of data pertaining to a large, complex product line is made possible by the Hands-On Network Environment (HONE) discussed in this paper. HONE provides on-line interactive support to marketing, systems, and administrative personnel, and, most recently, to customers. The evolution of HONE is presented. Discussed in detail are new HONE distributed processing capabilities now enabled under an advanced network architecture. In that environment, the processing power and data bases of HONE and other host systems will be interconnected and support the speed and processing autonomy of IBM Personal Computers as workstations.

Performance considerations for a distributed data processing system designed for high availability *by S. Agassi, p. 200.* The high-availability requirements of computerized systems that are needed to meet the objectives of the organization are being acknowledged more and more by the data processing community. The paper presents the planning process for a distributed data processing system designed to meet high availability requirements. This process was performed as a systems engineering activity in order to assess the feasibility of the presented approach, which was proposed to a customer.

Information System Model and Architecture Generator *by K. P. Hein, p. 213.* The advent of integrated, shared-data systems has made it increasingly necessary to address the application development process from the architectural and manufacturing perspective rather than from a build-as-you-go job shop viewpoint. Although the Business Systems Planning (BSP) methodology provides an enterprise-wide strategic Information Systems plan, it is still at an abstraction level that leaves the traditional gap between "requirements" and implementations untouched. The Information System Model and Architecture Generator (ISMOD) tool complements and enhances BSP by mechanizing the planning process, thus providing a facility to narrow this gap by allowing orderly and consistent top-to-bottom architectural decomposition of the enterprise environment. It is an enterprise planning vehicle and not an implementation system, but it is the first critical component to support an integrated systems architecture effort. It automates and, to a large extent, formalizes laborious requirements documentation process preceding code development, and it does this "top to bottom," from a global, enterprise-wide, information requirements viewpoint. This paper discusses the overall architectural concepts of integrated data systems development, the place of ISMOD within it, and the specific facilities, techniques, and information provided by the system.

A single-system interface using the IBM 3270-PC *by M. M. Ghiotti, p. 236.* Many businesses use a variety of terminal types connected to central host computers. Presented here is a rationale and the experience gained with a single terminal type—the IBM 3270-PC—interconnected with hosts via the Application Program Interface to achieve enhanced user efficiency.

Strategies for problem prevention *by J. Newton, p. 248.* A philosophy of preventing problems from occurring in a data processing installation rather than reacting to problems is becoming increasingly necessary. The institution of comprehensive and formally managed testing strategies is an important step in this direction. Such strategies are discussed, and it is shown that they also support disaster backup/recovery plans.

Customer Information Control System—An evolving system facility *by B. M. Yelavich, p. 264.* Presented is an overview of the present CICS architecture. Discussed is the evolution of that original design as a transaction management system that accommodates data base management, operating systems, and input an output devices as well as hardware of increasing numbers and complexity. User needs past and present are analyzed with a view toward understanding how CICS might evolve in the future.

An approach to high availability in high-transaction-rate systems *by R. C. Brooks, p. 279.* In business enterprises,

it is important that high availability be maintained in the computer systems used by the enterprises, particularly in systems that have high transaction rates. A way of maintaining high availability is discussed, including the implementation that should be undertaken and the design issues involved. Some additional steps for further improvements are also offered.

The System Planning Grid: A model for building integrated information systems *by B. R. Buckelew, p. 294.* Information systems have evolved as a result of technological advances and the increasing demand for information. Over the past few years, systems that developed separately are being forced to merge. This paper describes a model for building a set of integrated architectural guidelines to ensure that a "system" is being built. The use of the System Planning Grid as a model for setting product standards and organization responsibilities will also be discussed.

An information technology architecture for change *by M. W. Mudie and D. J. Schafer, p. 307.* This paper defines a technology architecture for information processing in large corporations. It describes a matrix of processing environments consisting of three processing types: production, decision support, and office; three processing locations: centralized, departmental, and workstation; and a methodology for implementing applications in those environments. Key to the architecture is a supporting framework comprising the communications network, a data service function, an office services function, enabling software, and support organizations. This approach is designed to provide an integrated information system to support organizations whose business environment is changing, and where flexibility, responsiveness to change, and cost effectiveness are vital. The approach is representative of methods used by systems engineers in assisting customers to decide on a system configuration that best suits their needs.

Cache-DASD storage design for improving system performance *by C. P. Grossman, p. 316.* This paper discusses three examples of a cache-DASD storage design. Precursors and developments leading to the IBM 3880 Storage Control Subsystems are presented. The development of storage hierarchies is discussed, and the role of cache control units in the storage hierarchy is reviewed. Design and implementations are presented. Other topics discussed are cache management, performance of the subsystem, and experience using the subsystem. It is shown that a cache as a high-speed intermediary between the processor and DASD is a major and effective step toward matching processor speed and DASD speed.

Volume 25, Number 1, 1986 *G321-0082*

The IBM 3090 system: An overview *by S. G. Tucker, p. 4.* The first part of this paper places the IBM 3090 system in historical perspective with respect to its predecessors. Treated briefly are the technology and the design process, both of which are critical to the development of the 3090. Presented in detail is the 3090 system itself, with emphasis on its features that differ from those of prior systems.

IBM 3090 performance: A balanced system approach *by Y. Singh, G. M. King, and J. W. Anderson, p. 20.* The IBM 3090 system represents the highest level of system performance offered by IBM to date. To realize the full performance potential of this system, it is essential to maintain a balance among its various components. The major components of the system are the processor(s), storage, I/O, and the software that manages the system resources. Their performance attributes are discussed and their effect on system performance illustrated by laboratory benchmark measurements for the MVS and VM operating systems.

Engineering and scientific processing on the IBM 3090 *by D. H. Gibson, D. W. Rain, and H. F. Walsh, p. 36.* The IBM 3090 processor implementation of the System/370 Vector Architecture represents a major new system design for engineering and scientific processing, featuring both scalar and vector capability in a uniprocessor and in a dyadic and four-way parallel processing environment. The history of large-scale scientific processing is reviewed, leading to a statement of current requirements. The design objectives for scalar, parallel, and vector capabilities are identified, followed by a summary of the resulting 3090 features. Selected highlights of the vector hardware are given, followed by a summary of the supporting software. The paper concludes with a discussion of performance, beginning with the identification of suitable applications. An example is given of one application utilizing each of the three capabilities: scalar, parallel, and vector. Several of the most important performance parameters are identified.

The IBM System/370 vector architecture *by W. Buchholz, p. 51.* Discussed is the instruction-set architecture of the IBM System/370 vector facility, a compatible extension of the System/370 architecture. Both the base system, which is a general-purpose System/370 processor, and the optional vector facility employ a register type of organization. Data formats are the same, arithmetic operations produce exactly the same results, arithmetic exceptions are handled in the same way, and instructions are precisely interruptible for page faults and other causes in the same manner as those of the base system. This approach permits substantially increased performance on vectorizable programs with only a modest increase in hardware and software, while retaining the ability to run existing nonvector programs unchanged.

Vector system performance of the IBM 3090 *by R. S. Clark and T. L. Wilson, p. 63.* Performance of the Vector Facility of the IBM 3090 processor is discussed. The paper has two parts, the first presenting factors affecting performance measurement of the Vector Facility and the criteria for its design. In the second part, use of the 3090 storage hierarchy to support the vector processing implementation is the main aspect of the discussion.

The System Usability Process for Network Management Products *by K. D. Gottschalk, p. 83.* This paper presents an overview of a process for system usability. The process is a systematic series of activities and procedures designed to improve the usability of software network management products. The elements of the process are given future directions for evaluating usability described.

Network management software usability test design and implementation *by L. C. Percival and S. K. Johnson, p. 92.* The approach used at one of IBM's development sites for usability testing is somewhat different from methods used elsewhere. The approach was developed specifically for testing of software communications products

as one aspect of the System Usability Process. The test design and implementation are described.

Improving availability of software subsystems through online error detection *by L. Koved and G. Waldbaum, p. 105.* A VM/370 program called Auditor detects faults in the operation of computer software subsystems and attempts to restore service as quickly as possible. Through a series of periodic tests, Auditor diagnoses whether these subsystems are operating properly. When faults are detected, service restoration procedures are automatically called, and the persons responsible for the subsystems are notified. The various types of faults are recorded for subsequent analysis.

Workstations and mainframe computers working together *by J. K. Kravitz, D. Lieber, F. H. Robbins, and J. M. Palermo, p. 116.* The history and design philosophy of a research project that produced a prototype software package are described. The package provides cooperation between large-scale mainframe computing systems and personal desktop workstations. Also discussed are the features of PC/VM Bond, the product that grew out of the research project.

Volume 25, Number 2, 1986 *G321-0083*

Introduction to IBM's knowledge-systems products *by A. J. Symonds, p. 134.* The industrialization of artificial intelligence is believed by many to be a technology that will contribute to a new generation of "smart" computer systems. Technical managers who are users or suppliers of computer systems are trying to understand how the technology can help them, and they are finding it an elusive subject to grasp. This introductory paper starts with a technology overview that aims to address this need for understanding and provide a suitable background for the papers that follow.

Knowledge-based systems in the commercial environment *by E. D. Hodil, C. W. Butler, and G. L. Richardson, p. 147.* Knowledge-based systems are among the first applications of artificial intelligence to make the crossover from the laboratory to the real-world commercial environment. Typically, artificial intelligence systems have been implemented in the LISP programming language on specialized hardware. The experimental nature of early systems has allowed many of them the luxury of having little or no interface to existing hardware, software, or data. In this paper, arguments are presented to demonstrate the feasibility of implementing knowledge-based systems using traditional hardware and software. Also, an architecture is proposed for knowledge-based shell systems that is compatible with the software development environment of large commercial information systems organizations. To demonstrate these concepts, an example system is shown.

YES/MVS and the automation of operations for large computer complexes *by K. R. Milliken, A. V. Cruise, R. L. Ennis, A. J. Finkel, J. L. Hellerstein, D. J. Loeb, D. A. Klein, M. J. Masullo, H. M. Van Woerkom, and N. B. Waite, p. 159.* The Yorktown Expert System/MVS Manager (known as YES/MVS) is an experimental expert system that assists with the operation of a large computer complex. The first version of YES/MVS (called YES/MVS I) was used regularly in the computing center of IBM's Thomas J. Watson Research Center for most of a year. Based on the experience gained in developing and using YES/MVS I, a second version (YES/MVS II) is being developed for further experimentation. This paper discusses characteristics of the domain of large computing system operation that have been illuminated by the YES/MVS I experience, and it describes the modifications in the design of YES/MVS II that are an outgrowth of the YES/MVS I experience.

The genesis of a knowledge-based expert system *by J. A. Voelker and G. B. Ratica, p. 181.* This paper discusses the genesis, development, and testing of a knowledge-based expert system called the Contract Support Services Consultant. This system aids in the process of estimating, bidding, and preparing agreements for certain services required for the rearrangement, relocation, discontinuance, and reinstallation of IBM equipment. The goals in developing the prototype of the Contract Support Services Consultant were to capture the knowledge and experience of experts and make them available to nonexperts in a convenient, consistent way. The result is a system that improves the accuracy, consistency, and timeliness of the estimates and bids.

Prolog for applications programming *by W. G. Wilson, p. 190.* This paper discusses the problems and benefits of using the Prolog language to write application programs. Much attention is currently focused on expert-systems shells, which play a role in artificial intelligence (AI) systems similar to that of application generators in more conventional applications. The Prolog programming language embodies many of the features found in these shells, while providing a relatively general and complete programming language. MVS performance tuning is used as an application that typifies a broad class of applications suitable for implementation in this language. Some of the difficulties that had to be overcome to use the language are presented, with their solutions.

The numeric representation of knowledge and logic–Two artificial intelligence applications in medical education *by W. D. Hagamen and M. Gardy, p. 207.* MEDCAT (medical diagnosis, consultation, and teaching) is a program that makes the diagnoses from empiric data stored in patient records, explains its reasoning in response to questions (consultant mode), and uses its logical and communicative skills to instruct medical students in the proper approach to medical diagnosis (student mode). MEDCAT's reasoning can be modified by free-format discussion with physicians. CATS (computerized anatomical teaching system) is an entirely separate program designed to teach gross anatomy. Like MEDCAT, it has a consultant mode that the student may use to explore the program's reasoning, and a student mode in which the program takes the initiative. A prominent feature of CATS is its ability to discover meaningful general principles that reduce the need for memorization. Despite important differences in the subject matter, the data structure and code are very similar in the two programs. Both use a powerful natural-language interface that parses the input and generates the output.

The Portable Inference Engine: Fitting significant expertise into small systems *by N. A. Burns, T. J. Ashford, C. T. Iwaskiw, R. P. Starbird, and R. L. Flagg, p. 236.* The Portable Inference Engine (PIE) is the nucleus of an expert system that allows the segmentation of rules in order to utilize large knowledge bases in limited memory. If the expert-systems rules can be divided into segments of highly related rules with little interaction among those segments, such knowledge-base segments can then be paged in and out of memory on demand. PIE gathers

information by querying the user and executing external procedures in order to conclude goals.

Computer processing of dates outside the twentieth century *by B. G. Ohms, p. 244.* This paper presents practical solutions to problems envisioned in extending computer processing of dates beyond the twentieth century. Many data processing managers are concerned with processing cross-century dates, and in doing so using existing systems, with a minimum of disruption to normal operations. The use of existing date formats can eliminate the need for massive system modifications. Methods of using existing date formats across century boundaries are explained. The use of a format termed the Lilian date format in honor of Luigi Lilio, the inventor of the Gregorian calendar, is introduced. The requirements for an effective date-processing algorithm are presented.

Volume 25, Number 3/4, 1986 *G321-0084*

Systems architecture in transition–An overview *by H. Lorin, p. 256.* Systems architecture refers to the distribution of function and control among elements of a system. It is primarily a structural concept that includes the original meaning of the word architecture in the form of "processor architecture." This paper understakes to describe topics of current interest in the evolution of computing structures. It discusses various unit structures that may emerge as the economics and capabilities of technology relax more and more constraints. Of particular interest is the internal structure of a cental computing complex, the relation of computing elements and I/O elements, and the maturity of the I/O elements. The paper also suggests that the structures found within a single computing unit may be realized across larger elements more widely dispersed. Hardware and software issues are addressed.

Impact of memory systems on computer architecture and system organization *by R. E. Matick, p. 274.* The largest part of computer architecture, in both the cental processing unit and the overall system, has been and continues to be directly influenced in one way or another by the types of memory systems available. This is readily apparent in certain areas such as I/O architecture and memory hierarchies. However, the pervasiveness of this influence throughout the entire system is not so obvious. This paper demonstrates this relationship and shows how it has affected computer architecture over the years. Two approaches are used, the first being a direct look at how specific architectures attempt to circumvent the limitations of the associated memory system. This includes such topics as the internal architecture of CPUs: memory hierarchies and virtual memory, I/O architecture, file structuring, and data base architecture. Second, a *gedanken* (thought) experiment is used to predict future trends. It is assumed that very large-scale integration will evolve to the point at which we can have nearly any main memory system we desire, with some reasonable constraints. The architectural changes that might take place will be seen to be precisely related to the weaknesses in current memory systems which various architectures currently attempt to circumvent.

Computing as a tool for human augmentation *by W. J. Doherty and W. G. Pope, p. 306.* The IBM Thomas J. Watson Research Center in Yorktown, New York, has experienced a factor of twenty times increase in the past ten years in the amount of time its people spend using computers interactively in their work. This is twice the penetration rate of television in the 1950s. A similar degree of penetration is expected to happen in the rest of industry in the next ten years. That will mean a major departure from traditional data processing, with computers being used as tools to augment the users' abilities in all phases of their work. Examples of human augmentation, as seen in the work of several researchers, are offered in this paper. The integration of large numbers of personal workstations into this environment has given us new understanding of how to work. The causes, impediments, and consequences of these changes are described, with emphasis on human requirements for bandwidth, response time, tools, on-line storage, and computing capacity.

IBM small-system architecture and design–Past, present, and future *by G. G. Henry, p. 321.* Small computer systems have become widespread and important parts of the computer industry. In this paper, a selection of IBM's small-system architectures and design approaches are reviewed. Then current architectures are discussed, with an emphasis on the RT Personal Computer. A brief presentation of trends in small systems is provided.

Software engineering: An emerging discipline *by R. Goldberg, p. 334.* Software engineering is an emerging discipline whose goal is to produce reliable software products in a cost-effective manner. This discipline is evolving rapidly as the challenges faced by its practitioners keep extending their skills. This paper gives a quick tour of the main ideas and thrusts that have driven software engineering in its first 25 years and attempts to look ahead at the next set of advances.

Tools for building advanced user interfaces *by J. L. Bennett, p. 354.* System developers are noticing that their design decisions strongly affect computer usability. The design of the user interface has an important bearing on the knowledge users must have to accomplish work through the user-computer interface. Recognition of this fact is leading to the development of User Interface Management Systems (UIMSs). A UIMS is a design concept for separating the details of user interaction from the details of advanced applications. This paper shows how UIMS research and research into the representation of user process knowledge (i.e., user how-to-do-it skills) can help developers understand issues involving ease of learning and ease of use. This parallel progress in UIMS development and in user modeling makes it easier to build high-quality advanced user interfaces.

Open Systems Interconnection *by J. R. Aschenbrenner, p. 369.* The subject of Open Systems Interconnection (OSI) standardization is becoming increasingly important to the telecommunications and information processing communities. A number of OSI standards have been completed, others are near completion, and initial product offerings by vendors have begun. This paper briefly defines what OSI is, the interrelationships of the various standards bodies, and the goals and benefits to users, vendors, country post telephone and telegraph bodies, common carriers, and governments. The IBM view of OSI and how it relates to Systems Network Architecture is also discussed.

An advanced voice/data telephone switching system *by J. M. Kasson, p. 380.* This paper describes a voice/data circuit-switching system known as the ROLM CBX II. The paper first discusses what the ROLM CBX II family

does, describing the most important functions and relating them to popular voice and data applications. The second section describes how the CBX II works, delineating the architecture and giving some details of the system implementation. The final section offers an assessment of what the CBX II and similar products might become in the future.

The evolution of printers and displays *by A. F. Mayadas, R. C. Durbeck, W. D. Hinsberg, and J. M. McCrossin, p. 399.* Printer and display technologies have undergone remarkable changes since the beginning of the computer era. In this paper we trace the evolution of these two types of I/O devices, from the middle of the 1940s to the present, and show how computer system evolution has influenced the designs and technologies of I/O devices.

Volume 26, Number 1, 1987 *G321-0085*

Structures for networks of systems *by A. L. Scherr, p. 4.* This paper describes how systems will be interconnected in the future, the roles that they will play, and the trade-offs that affect these roles. Starting with a general model for structuring a network of systems, general trade-offs in cost and performance are discussed relative to where functions are placed in the network. Several general principles for data and function placement in a network of systems are derived from these trade-offs. The optimal roles for each of several layers of a network of systems are discussed. Finally, conclusions are drawn regarding the design of future networks of systems.

SNA: Current requirements and direction *by R. J. Sundstrom, J. B. Staton III, G. D. Schultz, M. L. Hess, G. A. Deaton, Jr., L. J. Cole, and R. M. Amy, p. 13.* Since its announcement in 1974, Systems Network Architecture (SNA) has evolved in terms of its functional content, configurational flexibility, and network management services. This paper briefly traces this progress to the present and examines the more recent advances in greater detail. It then discusses known requirements for enhanced application and transaction services, for additional provisions for very large networks, for continuing adaptation of small-system and transmission media advances, for inclusion of additional management capabilities, and for further integration of network standards—all of which will shape future SNA developments.

Prospects and design choices for integrated private networks *by P. E. Green, Jr. and D. N. Godard, p. 37.* This paper reviews some of the choices that will be available in the next few years, as the much-discussed move toward implementing voice and data integration within a single wide-area integrated private network proceeds. After the term *wide-area integrated private network* has been defined, a discussion is given of requirements the network ought to satisfy for its users. Then two particularly promissing approaches, fast packet switching (FPS) and hybrid switching (HS), are defined, and specimen design points for FPS and HS are postulated, so that the two can be compared. While a definitive comparison would require systematic cost and performance studies, much insight can be gained from the qualitative comparison that we present here. We assess some of the arguments that have been put forward in favor of FPS or HS and conclude that, although today both architectures have promise, and research on both should continue, FPS appears to be slightly simpler to implement and operate.

Robotics *by J. U. Korein and J. Ish-Shalom, p. 55.* This paper is a survey that is intended to give the reader an introduction to some issues and problems in the field of robotics today. The first section discusses industrial applications of robotics and the requirements they engender. A substantial section is included on robot programming, including programming languages, motion programming, and techniques. This is followed by a section on trajectory planning. Issues in both robot-level trajectory planning and task-level trajectory planning are discussed. The section on control is divided into three parts: controller objectives, the system model, and controller types. Very brief discussions of actuators, sensing, and end effectors are also included.

Database technology *by P. G. Selinger, p. 96.* Computers were originally invented and used to ease and automate the task of computation. As the word "computer" implies, these early machines were used for calculations, such as tabulating census data. As a side effect, the technology needed for storing data was also invented to provide the computational engine with input data and allow it to output results. This means of permanently storing data included punched cards, tape, and disks. Throughout the 1950s and most of the 1960s, the management of stored data was done as required; file systems stored data according to user-defined formats and kept a table of contents. Users shared data by equally *ad hoc* means, generally by taking turns accessing the same device. Over the years, database technology has evolved through at least three generations to a diverse and sophisticated set of data management tools, as discussed in this paper. This paper has three major sections. Presented first is an introduction to database technology. Presented next is a description of the evolution of database technology from early computing to the sophisticated systems of today. The third section presents a view of both the driving forces that will influence the database technology of the future and also the resulting new directions for the future.

A perspective on the 801/Reduced Instruction Set Computer *by M. E. Hopkins, p. 107.* From the earliest days of computers until the early 1970s, the trend in computer architecture was toward increasing complexity. This complexity revealed itself through the introduction of new instructions that matched the application areas. Microcode was an implementation technique that greatly facilitated this trend; thus, most computers were implemented using microcode. In 1975, work began at the Thomas J. Watson Research Center on an experimental minicomputer. This project, termed the 801 project, questioned the trend toward complexity in computer architecture. It was observed that most of the complex instructions were seldom used. Thus, a computer could be designed with only simple instructions without drastically increasing the path length or number of instructions required to implement an application. This made it possible to implement a machine without resorting to microcode, which improved performance. This paper described the background and evolution of these ideas in the context of the 801 experimental minicomputer project.

Data communications: The implications of communication systems for protocol design *by B. C. Goldstein and J. M. Jaffe, p. 122.* The construction of a communication

network architecture, specifying protocols by which systems communicate, is a complex art. Much has been written about the optimal protocols for theoretical models of systems. This paper points out that protocol design must depend on the "nuts and bolts" of the systems which implement the protocols. Numerous examples are provided to support this thesis. The paper also briefly discusses other issues that influence protocol design and draws lessons for standards activities.

A large-scale computer conferencing system *by D. M. Chess and M. F. Cowlishaw, p. 138.* This paper discusses the relationships between computer-mediated communications and other forms of communication and describes a particular computer conferencing system in use within IBM. The system described is quite large, with over three thousand contributors and over twenty thousand readers. We discuss the structure of the system, the actions that users can take, and the ways in which the system is being used. Neither the definitions presented nor the system described are intended to be the last, or only, word on the subject; as computer-mediated communications and distribution become more and more important in the business and professional communities, we will need more ways of thinking about communication systems and about information distribution in general.

Volume 26, Number 2, 1987 *G321-0086*

OSI-SNA interconnections *by K. K. Sy, M. O. Shiobara, M. Yamaguchi, Y. Kobayashi, S. Shukuya, and T. Tomatsu, p. 157.* As Open Systems Interconnection (OSI) becomes an international standard, it is gaining support in both industry and government agencies. One of the major applications of OSI is to act as an intermediary between heterogeneous networks. This paper discusses a scheme for interconnecting a Systems Network Architecture (SNA) network with OSI. This scheme is based on a joint study between IBM Japan and Nippon Telegraph and Telephone Corporation conducted during 1984. Fundamental relationships between OSI session and transport layers and SNA Logical Unit type 6.2 are explored. An OSI-SNA gateway structure is examined, and data units, address translation, and exception handling are discussed.

Visual interpretation of complex data *by E. J. Farrell, p. 174.* With increasingly complex digital simulations and computations, larger volumes of output are generated, and users must select a concise method of displaying the output and extracting relevant information. A set of imaging functions and display modes is developed to interpret data effectively for a wide range of applications. The imaging functions are complementary. Each function is useful for a different aspect of data interpretation. The relationships between variables and the global structures within the data are obtained with different display modes such as multiple windows and animation. With this set of complementary imaging functions and display modes, more information is obtained than with prior imaging methods. Also, more complex simulation studies are feasible since the results can now be visualized.

An incidence-matrix-driven panel system for the IBM PC *by P. Halpern, S. M. Roberts, and L. Lopez, p. 201.* A set of programs called TRYLON is discussed which permits the application developer to design and create a set of intelligent full-screen panels. These panels serve as a user interface for application programs. The panels and the linkages between them are uncoupled from the application code, thereby reducing programming effort and development time. An incidence matrix is used to describe the graph of a network composed of the panels. Because the paths among the panels are specified by entires in the incidence matrix at panel creation time, there is no need to program the logical constructs for the network.

A page-swapping prototype for VM/HPO *by W. H. Tetzlaff, T. Beretvas, W. M. Buco, J. Greenberg, D. R. Patterson, and G. A. Spivak, p. 215.* This paper discusses a series of changes that were made to a system running the Virtual Machine/System Product with the High Performance Option to enhance paging. The motivation and the background for these enhancements are discussed, and the design of a series of experimental paging subsystems is described and contrasted with the old design: specifically, the new algorithms for main memory management, block paging, working set identification, trimming, prepaging, page replacement, page-out device selection, and page-out slot selection. The performance impact of these changes is illustrated by results of benchmark measurements, which are then contrasted to measurements without the enhancements. Some things learned in running the prototype are discussed and conclusions drawn.

Volume 26, Number 3, 1987 *G321-0087*

Message-handling systems based on the CCITT X.400 recommendations *by T. E. Schütt, J. B. Staton III, and W. F. Racke, p. 235.* Message-handling systems allow the exchange of electronic mail between computers. The International Telegraph and Telephone Consultative Committee (CCITT) has proposed a standard for message-handling systems in the form of the X.400 series of recommendations that has been widely recognized by computer manufacturers and communications carriers. This paper provides a tutorial on the X.400 recommendations and then describes two prototypes developed by the IBM European Networking Center in Heidelberg, Germany, in cooperation with its research partners. The prototypes were demonstrated together with X.400 prototypes from other manufacturers at the CeBIT 86 trade fair in Hannover, Germany.

Specification and implementation of an ISO session layer *by A. Fleischmann, S. T. Chin, and W. Effelsberg, p. 255.* This paper describes a novel technique for the specification and implementation of layered communication software. The technique is called Parallel Activity Specification Scheme (PASS) and is based on an extended-state machine model of protocol automata. It allows a convenient description of the communication behavior of concurrent systems and semiautomatic generation of programming language code from the specification. The first large-scale experience gained with this technique was in the specification and implementation of an ISO session layer. The code generation process and the embedding of the session code into a portable OSI operating system environment are described in detail.

A framework for information systems architecture *by J. A. Zachman, p. 276.* With increasing size and complexity of the implementations of information systems, it is necessary to use some logical construct (or architecture)

for defining and controlling the interfaces and the integration of all of the components of the system. This paper defines information systems architecture by creating a descriptive framework from disciplines quite independent of information systems, then by analogy specifies information systems architecture based upon the neutral, objective framework. Also, some preliminary conclusions about the implications of the resultant descriptive framework are drawn. The discussion is limited to architecture and does not include a strategic planning methodology.

The structure of System/88, a fault-tolerant computer *by E. S. Harrison and E. J. Schmitt, p. 293.* In recent years, there has been a growing requirement for continuous processing capability approaching 24 hours per day, 7 days per week. Industries such as finance, transportation, securities, and telecommunications have continuous-availability requirements that can approach downtimes of not more than three minutes per year. This paper describes configurations of the Stratus/32 continuous processing computer system that are marketed as the IBM System/88 through an agreement with Stratus Computer, Inc. The system achieves its fault tolerance via hardware duplexing coupled with a distributed operating system that allows system resources to be distributed over many separate computers while maintaining a single systems image to the end user. This single systems image may also be extended across a network of multiple systems. The way in which software makes this distribution possible and the way in which system resources are named to achieve transparent distribution across the system are described in the paper. Also described are the transaction processing services that are part of the operating system and allow transaction programs to be written to operate effectively over the distributed system, by means of a requester-server structure approach.

Volume 26, Number 4, 1987 *G321-0088*

Advanced Interactive Executive (AIX) operating system overview *by L. K. Loucks and C. H. Sauer, p. 326.* The Advanced Interactive Executive (AIX) is the operating system used in the RT Personal Computer. It is a portable operating system architecture that is suitable for a wide range of computer architectures and customer requirements. Discussed in this paper are the structure and services of AIX.

The IBM RT PC ROMP processor and memory management unit architecture *by R. O. Simpson and P. D. Hester, p. 346.* The ROMP processor is the microprocessor used in the IBM RT PC. It is a 32-bit processor with an associated memory management unit implemented on two chips. ROMP is derived from the pioneering RISC project, the 801 Minicomputer at IBM Research. This paper describes some of the trade-offs which were made to turn the research project into a product. It gives an introduction to the architecture of ROMP, including the addressing model supported by ROMP's memory management unit. Some of the unique features of the programming model are explained, with high-level language coding examples which show how they can be exploited. ROMP's architecture is extensible, and the fact that almost all programming for the RT PC has been in high-level languages means that the RT PC hardware architecture can be extended as needed to meet future requirements while preserving the investment in existing software.

Advanced Interactive Executive program development environment *by R. Q. Cordell II, M. Misra, and R. F. Wolfe, p. 361.* The IBM RT Personal Computer uses the Advanced Interactive Executive as an operating system. This operating system provides a distinct environment for the development of programs. Some of the characteristics of application development with this operating system, some of its features that influence application design, and the basic program development tools are described.

AIX usability enhancements and human factors *by F. C. H. Waters, R. G. Bias, and P. L. Smith-Kerker, p. 383.* As microcomputers become capable of running increasingly large and complex operating systems, the question of the usability of those operating systems becomes critical. Most microcomputer users neither are nor want to be systems programmers, yet most of the existing large operating systems assume the existence of a dedicated systems programming organization to install and maintain system software. This paper describes the process by which a large existing operating system was modified to allow it to be installed, configured, maintained, and used by individuals with minimal programming knowledge. We describe the aspects that had to be changed, the kinds of modifications that were required, the reasoning behind those modifications, and the priorities that constrained our activity. We also describe the development process by which potential usability problems were identified and corrections were defined, implemented, and validated.

Box structured information systems *by H. D. Mills, R. C. Linger, and A. R. Hevner, p. 395.* The box structure methodology for information systems development is based on a usage hierarchy of data abstractions, in which each abstraction is defined in three distinct forms, called its *black box*, its *state machine*, and its *clear box*. Each of these three box structures defines identical external behavior, but with increasing internal visibility, to provide a hierarchical structure which supports the systems development principles of referential transparency, transaction closure, state migration, and common services. This hierarchy of box structures provides, in turn, a basis for orderly management of information systems development by a finite set of analysis and design tasks in a spiral development process. The methodology and its use are described.

A perspective on Advanced Peer-to-Peer Networking *by P. E. Green, Jr., R. J. Chappuis, J. D. Fisher, P. S. Frosch, and C. E. Wood, p. 414.* This paper is intended to familiarize the reader with the many reasons for undertaking the design and implementation of peer networking on small and intermediate business machines such as the IBM System/36 family. Such networking function was recently announced as IBM's Advanced Peer-to-Peer Networking (APPN) on Release 5 of the System/36. This paper sets the stage for a companion paper in this same issue, which discusses the implementation experience and details of the System/36 APPN product. In the present paper, the history of System/36 communication is first reviewed, and it is shown how APPN was a natural evolution from earlier function. Then an extensive study of user requirements that was started in 1982 is summarized. The paper concludes with

a brief technical tutorial on the structure of the APPN design.

Implementing System/36 Advanced Peer-to-Peer Networking *by R. A. Sultan, P. Kermani, G. A. Grover, T. P. Barzilai, and A. E. Baratz, p. 429.* System/36 Advanced Peer-to-Peer Networking (APPN) provides highly dynamic, fully distributed peer networking for low-end processors. It is built upon existing SNA Logical Unit 6.2 and Node type 2.1 support. APPN presents System/36 users with a simplified model of communications. The structure of the APPN subsystem is outlined, with particular emphasis on the integration of APPN functions with existing SNA support. The authors describe how particular aspects of the APPN design have been tuned to the System/36 operating environment.

Volume 27, Number 1, 1988 *G321-0089*

SNA network management directions *by D. B. Rose and J. E. Munn, p. 3.* Network management is the process of monitoring and controlling the components of a communication-oriented network of information systems in the areas of configuration management, operational control, problem management, change management, and performance and accounting management. This paper discusses the evolution of the SNA network management architecture and products that implement that architecture, and describes their likely future direction.

Utilizing the SNA Alert in the management of multivendor networks *by R. E. Moore, p. 15.* Managing multivendor networks is one of the largest challenges facing vendors and customers in data processing and telecommunications. This paper focuses on one aspect of managing multivendor network environments: problem notification, isolation, and resolution, via Systems Network Architecture's Alert. It describes an extension to the SNA Alert function, termed the generic Alert, that makes it possible for various vendors' products, as well as customer-written applications, to send Alerts of the same type to a single Alert receiver. It also describes IBM's implementation of the Alert receiver for the System/370, the NetView™ program product. Among the facilities that the generic Alert architecture provides to an Alert sender are the following: (1) code points that index short descriptions of Alert conditions, probable causes of these conditions, and recommended operator actions; and (2) vehicles to carry product-unique text. This text can be used for further characterizing an Alert condition or specifying a particular operator action.

NetView/PC *by M. Ahmadi, J. H. Chou, and G. Gafka, p. 32.* NetView/PC™ is an IBM program offering that provides the first implementation of a Systems Network Architecture (SNA) service point on an IBM Personal Computer. It allows integration of non-SNA devices such as computerized branch exchanges or Token Ring local-area networks into a central SNA network management facility by forwarding Alert information from these devices to a host-based network management product such as NetView™. It also provides some focal point services, including logging and display of Alerts and problem creation and tracking. Non-IBM products may use NetView/PC services via a vendor application programming interface (API).

An integrated network management product *by D. Kanyuh, p. 45.* The NetView™ program is the cornerstone of IBM's network management concept. It allows enterprises to consolidate their network operations at a centralized point and is a key element in providing the function to perform the major disciplines of network management. This paper describes how and why the NetView program originated, the components that make up the product, the network management functions provided by these components, such as operations, problem management, and performance management, and the contribution of the NetView program to the open network management direction.

An architecture for a business and information system *by B. A. Devlin and P. T. Murphy, p. 60.* The transaction-processing environment in which companies maintain their operational databases was the original target for computerization and is now well understood. On the other hand, access to company information on a large scale by an end user for reporting and data analysis is relatively new. Within IBM, the computerization of informational systems is progressing, driven by business needs and by the availability of improved tools for accessing the company data. It is now apparent that an architecture is needed to draw together the various strands of informational system activity within the company. IBM Europe, Middle East, and Africa (E/ME/A) has adopted an architecture called the E/ME/A Business Information System (EBIS) architecture as the strategic direction for informational systems. EBIS proposes an integrated warehouse of company data based firmly in the relational database environment. End-user access to this warehouse is simplified by a consistent set of tools provided by an end-user interface and supported by a business data directory that describes the information available in user terms. This paper describes the background and components of the architecture of EBIS.

Volume 27, Number 2, 1988 *G321-0090*

The design of Operating System/2 *by M. S. Kogan and F. L. Rawson III, p. 90.* The design of Operating System/2™ (OS/2™) is a result of matching the requirements of IBM and its customers for a new operating system for various models of the Personal System/2® with the need for continuity with a very large body of established DOS applications. The design of OS/2 represented a significant challenge both in meeting these requirements and in making efficient use of the hardware. In this paper, the design characteristics of OS/2 are discussed.

OS/2 EE Database Manager overview and technical highlights *by P. Y. Chang and W. W. Myre, p. 105.* Structured Query Language (SQL) has become an industry standard. It is supported by mainframe products. This paper describes the OS/2 EE Database Manager, which is based on the relational database model of E. F. Codd and on the SQL query language. A functional overview of the OS/2 EE Database Manager and OS/2 EE is provided; technology applied to different areas is highlighted.

OS/2 Query Manager overview and prompted interface *by S. L. Watson, p. 119.* Operating System/2™ (OS/2™) Query Manager provides a user interface for both novice and sophisticated database users of the OS/2 Database Services. It offers defaults and standard options for the novice user. Prompting provides access

to the database without requiring extensive knowledge of Structured Query Language (SQL), yet it also allows the advanced user to completely customize screens and reports. Direct keying of SQL statements is allowed as a fast path for the knowledgeable SQL user. Functions of OS/2 Query Manager are described, including details of the user interface.

Writing an Operating System/2 application *by R. L. Cook, F. L. Rawson III, J. A. Tunkel, and R. L. Williams, p. 134.* This paper illustrates use of the key facilities of Operating System/2™ (OS/2™). It provides some guidance on how to use the interfaces and functions implemented by the system and then introduces the program development environment. Two examples demonstrate the use of some of the more interesting capabilities. The paper discusses many of the significant differences between the functions of OS/2 and those of the Disk Operating System (DOS).

COBOL/2: The next generation in applications programming *by R. Sales, p. 158.* IBM COBOL/2 is a new compiler and debugger system for the Personal System/2® product range developed by Micro Focus Group PLC of the United Kingdom. In this paper, Robert Sales, a software development manager for Micro Focus who was instrumental in creating the COBOL/2 system, describes how COBOL/2 breaks new ground in providing support for many disparate COBOL language dialects and standards, as well as in providing support for OS/2™ on the Personal Computer architecture.

Understanding device drivers in Operating System/2 *by A. M. Mizell, p. 170.* To meet its design goals for multitasking, Operating System/2™ requires a device driver architecture for interrupt-driven device management. A device driver in OS/2™ is affected by the new architecture both in its structure and in its relationship to the system. An OS/2 device driver contains components, such as the Strategy Routine and Hardware Interrupt Handler, which have well-defined responsibilities. The basic form of these components is a FAR CALL/FAR RETURN model. The operating system calls the device driver components to handle certain types of events, such as an application I/O request or a device interrupt. In responding to these events, an OS/2 device driver must cooperate with the operating system to preserve system responsiveness by helping to manage the multitasking of concurrent activities. Since OS/2 uses both the real mode and the protected mode of the system processor to support DOS and OS/2 applications, respectively, the components of an OS/2 device driver must execute in both modes. In this manner, an OS/2 device driver can be viewed as an installable extension of the Operating System/2 kernel. Comparisons between IBM Personal Computer DOS and Operating System/2 are drawn to illustrate differences between device management and device driver architecture.

VGA–Design choices for a new video subsystem *by S. Thompson, p. 185.* The VGA (Video Graphics Array) video subsystem is provided as standard on the system boards of the IBM Personal System/2® Models 50 and above. VGA was designed to meet the objectives set for these new systems and to support compatibility with older IBM offerings, while at the same time providing greater performance and increased function. The IBM Enhanced Graphics Adapter (EGA) was chosen as the compatibility base for VGA, since EGA had become the video standard for IBM-compatible computer systems. Six new modes of operation were designed to meet the needs of new business and consumer applications and to improve the ergonomics of the systems. Higher-performance video presents several design problems, including electromagnetic interference, physical design size, and cost. These design problems were contained by implementing the VGA function in a single-gate array and by using an analog display interface. The use of a video digital-to-analog converter (DCA) allows the VGA subsystem to show any color from a choice of 256K colors when a color display is used, or 64 gray shades when a monochrome display is used. The VGA subsystem was designed to provide a uniform interface for color and monochrome that allows a color mode to be selected when a monochrome display is used, or a monochrome mode to be used on a color display. A color-summing algorithm was designed and implemented in the BIOS (Basic Input/Output System) software that will allow colors to be shown as shades of gray on the monochrome display.

The Realtime Interface Co-Processor Multiport/2 adapter *by S. C. G. Sykes, p. 198.* The Realtime Interface Co-Processor Multiport/2 is a programmable, multifunction adapter that extends the processing capabilities of the Personal System/2® and provides a solution to applications with unique communications requirements. Customized for speed and flexibility, the Multiport/2 is fully programmable and supports asynchronous, byte-synchronous, and bit-synchronous protocols on its eight communications ports. This powerful single-slot computer can handle functions that previously required processing by the PS/2®. Microcode on the Multiport/2 provides a realtime multitasking base on which custom applications can be built. This paper describes the Multiport/2, its microcode, system support software, and development tools.

An introduction to typographic fonts and digital font resources *by A. W. Griffee and C. A. Casey, p. 206.* Type has evolved from blocks of wood or metal bearing the raised character shape to the many and varied digitized representations of the character that are available through computer system technology. Typography is the art or technique of composing printed material from type. The evolution of digital type into the computers of today has opened the door of typography to people who have had little or no previous knowledge of the subject. It has also introduced a higher level of complexity to document composition and presentation service software than was previously required. Discussed in this paper are the art of composing printed material, the selection of an appropriate type design for a given application, the information required to create and manage a digital font resource, and the computer system's use of digital font resources to produce typographic-quality documents. These matters are examined in a way that introduces the reader to typographic fonts, the additional complexities involved, and the need for consistency in the definition and application of digital font resources.

Advanced Function Printing: A tutori. *. by R. K. deBry and B. G. Platte, p. 219.* Advanced Function Printing (AFP) is an IBM product for printing mixed text, image, and graphics in a system-printing environment. Described is the AFP printing model. We demonstrate the way in which this model is used for existing printing applica-

tions, enhanced line printing, and full advanced-function printing.

Architectures of Advanced Function Printing *by R. K. deBry, B. G. Platte, C. L. Berinato, and J. W. Marlin, p. 234.* Discussed is the use of the capabilities of all-points-addressable laser (page) printers in applications involving pages composed of text, image, and vector data in a device-independent way. Also preserved is the ability to describe and print complex documents composed of multiples of such pages. Provision is made for the migration of current line-printer applications to print using these new page printers. Three architectures are described that—along with an Advanced Function Printing (AFP) model—support these capabilities. Each of these architectures is described in the context of the current implementation of the Advanced Function Printing software.

Volume 27, Number 3, 1988 *G321-0091*

Introduction to Systems Application Architecture *by E. F. Wheeler and A. G. Ganek, p. 250.* Systems Application Architecture is a framework in which applications are developed so that they run consistently on major IBM computing systems. This paper presents the motivation and requirements for this framework and describes the main elements of its structure. It also discusses the effect on current processing technologies and on application development.

Common Communications Support in Systems Application Architecture *by V. Ahuja, p. 264.* Application execution in a Systems Application Architecture (SAA) network depends on the underlying capability of the network to obtain reliable connectivity and orderly data exchange among its system components. The objectives of SAA are distributed applications, distributed processing, and distributed data, which are achieved through interconnected SAA systems supporting appropriate interfaces and architectures. The Common Communications Support of SAA affords this capability by utilizing a number of Systems Network Architecture communication architectures and international standards. These architectures provide useful data interchange within SAA components by providing services ranging from managing data links to specifying data streams for user applications. This paper discusses the role of Common Communications Support and the means for SAA users to access this support, and provides an overview of the functions and roles of various component architectures of Common Communications Support, along with their interrelationships.

Common User Access–A consistent and usable human-computer interface for the SAA environments *by R. E. Berry, p. 281.* Systems Application Architecture (SAA) will allow customers to apply their investments in their computer operations across IBM's three major computing environments that exhibit unique characteristics in terms of architecture, workstations, operating systems, and system services. User experience is one of these investments. The Common User Access (CUA) establishes a degree of standardization that is compatible with the differences in the three environments and that supports transfer of users' experiences. CUA is based on a user-interface architecture that identifies fundamental elements of structure. The intent is to provide a transfer of users' conceptual-level learning across different and evolving technologies. CUA specifies user-interface components and guidelines to be used by application designers, and it provides a basis for programming development tool specifications.

Application enabling in SAA *by D. E. Wolford, p. 301.* The Common Programming Interface (CPI), one of the four key elements of Systems Application Architecture, comprises a growing set of programming languages and services. The CPI indirectly offers end-user access through the Common User Access by providing the application developer with the necessary interfaces. The CPI addresses the application development requirement for portability of applications and programmer skills. As the CPI continues to expand, it addresses the requirements for access to host data through intelligent workstations and for transparent access to remote data and applications.

Enabling the user interface *by S. Uhlir, p. 306.* Presenting a consistent interface to the user is one of the objectives of Systems Application Architecture (SAA). The development of SAA applications is simplified by providing enabling interfaces which help an application developer support the SAA user interface. Rather than providing a single-level enabling interface, SAA offers a spectrum of levels spread over two interfaces: the SAA Presentation Interface and the SAA Dialog Interface. This gives the application developer the freedom to choose the appropriate level of interface for the application.

Integrating applications with SAA *by L. A. Buchwald, R. W. Davison, and W. P. Stevens, p. 315.* Advances in computing technology and reductions in development cost have greatly increased the number of people who use computers, and have expanded the number and types of applications available to them. People want their applications to share data and to be consistent with one another with respect to terminology and appearance. They also frequently need access to applications and data on computers in other locations; the computers may be models and types that these persons do not normally use. Integrating application functions in a seamless environment is an important step toward satisfying some of these requirements. This paper discusses what integrated applications are, why they are valuable, and how Systems Application Architecture (SAA) can make it easier to develop them.

Designing SAA applications and user interfaces *by W. P. Dunfee, J. D. McGehe, R. C. Rauf, and K. O. Shipp, p. 325.* This paper describes a framework for developing applications that conform to Systems Application Architecture (SAA). The paper shows a high-level approach to creating a design; it gives examples of early modeling work with the user interface; and it appraises SAA through the eyes of several system designers. The usability of user interfaces has been evaluated through the modeling of office tasks. That experience is described, showing the influence of the SAA Common User Access (CUA) on the model and the influence of the model on CUA. Discussed is a design for distributed applications that fit within the SAA framework and the influence of SAA on the design of integrated distributed applications.

Distributed files for SAA *by R. A. Demers, p. 348.* Files are still a major way of storing data in computer systems, and they are a significant part of the information to be handled by the distributed processing networks that are developing. Systems Application Architecture is sup-

porting distributed files. In this paper, the goals, benefits, and problems of providing this support are discussed, along with the role of Distributed Data Management architecture.

Distributed database for SAA *by R. Reinsch, p. 362.* This paper describes, in general terms, distributed database and its relationship to Systems Application Architecture (SAA). It shows the importance to effective distribution of IBM's Structured Query Language (SQL), the database element of the Systems Application Architecture Common Programming Interface (SAA CPI). The paper defines five levels of distribution, showing how each fits real-world application requirements. Finally, it outlines the magnitude of the task.

SAA distributed processing *by A. L. Scherr, p. 370.* Discussed are motivations for distributed versus centralized data processing, the relative advantages of each, and the trade-offs involved as they relate to Systems Application Architecture (SAA). Presented is a taxonomy of the various approaches to designing applications to operate in a distributed manner. SAA support for these modes is described. The management of an enterprise-wide network of systems is discussed.

The Cross System Product application generator: An evolution *by W. K. Haynes, M. E. Dewell, and P. J. Herman, p. 384.* An application generator is a generalized application development tool used by professional programmers to develop applications using a fourth-generation language. This paper describes the requirements that led to the Cross System Product application generator, and how the product progressed from a single-environment product to the current multienvironment product. Also described are how the Cross System Product fits within Systems Application Architecture and how that may affect the future of the Cross System Product.

Volume 27, Number 4, 1988 *G321-0092*

IBM's directions in technical computing *by P. L. Prairie and A. H. Weis, p. 393.* Technical computing comprises hardware systems, software, tools, communications networks, and applications to significantly increase productivity and competitiveness. A Technical Computing Structure (TCS) is described which provides a framework and IBM's direction to integrate these elements, including both IBM offerings and industry standards to support the technical end user. Also discussed are some of the special studies undertaken to improve development programs for this environment.

Engineering and Scientific Subroutine Library for the IBM 3090 Vector Facility *by J. McComb and S. Schmidt, p. 404.* The Engineering and Scientific Subroutine Library (ESSL) provides FORTRAN, Assembler, and APL2 application programmers with a high-performance set of mathematical subroutines which take advantage of the performance gains offered by the IBM 3090 Vector Facility. This paper describes the contents of ESSL and presents some of the techniques that were used to develop high-performance vector subroutines. Other key design considerations such as accuracy, ease of use, and error handling are also discussed. This information should be useful to anyone developing programs for the IBM 3090 Vector Facility.

IBM Parallel FORTRAN *by L. J. Toomey, E. C. Plachy, R. G. Scarborough, R. J. Sahulka, J. F. Shaw, and A. W. Shannon, p. 416.* IBM Parallel FORTRAN is a compiler and library for writing and executing parallel programs. It provides language extensions for explicitly programming in parallel, and it also provides compiler enhancements for automatically generating both parallel and vector code. Parallel FORTRAN offers a language for parallel programming that is independent of the machine configuration and the operating system. The combination of Parallel FORTRAN and IBM 3090 multiprocessors can provide a significant reduction in turnaround time for applications.

Program locality of vectorized applications running on the IBM 3090 with Vector Facility *by K. So and V. Zecca, p. 436.* An instruction-level simulator is used to study the program locality of large scientific applications. The simulator, which models an IBM 3090 processor with Vector Facility and a cache, was developed to help a programmer improve the performance of an application through better understanding and use of the Vector Facility and the memory hierarchy of the IBM 3090 system. Our main observations on a set of scientific applications are as follows: (1) although the applications have different characteristics of memory accesses and vectorization, their program locality is high enough to take advantage of conventional cache structures; (2) the cache hit ratio of the vector execution can be quite different from (but not significantly lower than) that of the scalar execution of the same application; and (3) the application programs that are written to optimize the use of the memory hierarchy in the system generally result in higher cache hit ratios than the others. The cache performance of these applications with respect to various cache parameters is also presented. In particular, our study finds that the cache structure of the IBM 3090 is well suited for large scientific applications.

Programming style on the IBM 3090 Vector Facility considering both performance and flexibility *by H. Samukawa, p. 453.* To obtain high performance from the IBM 3090 Vector Facility, we must investigate vector instruction constructs in terms of the loop context of the application algorithm. We exemplify the method by linear algebra subroutines for basic matrix operations and a linear equation solver. In these examples, we clarify the mathematical meaning that each loop is computed by analyzing the loops in terms of a generic algorithm. This analysis helps us to achieve optimal loop selection. We then obtain additional performance gain by considering cache capacity. These procedures suggest that there are three levels of performance classification. They also show that program structure yields great benefits in terms of performance and generality of the program.

ICAP 3090: Parallel processing for large-scale scientific and engineering problems *by E. Clementi, D. Logan, and J. Saarinen, p. 475.* Described is the ICAP/3090 (for loosely coupled array of processors) parallel processing system. General parallel processing performance issues that determine the success of all multiple-instruction/multiple-data-stream parallel computing systems are examined in the context of large-scale scientific and engineering problems. Experiments with previous ICAP parallel processing systems that have made possible the present design of ICAP/3090 are also described.

Seismic computations on the IBM 3090 Vector Multiprocessor *by A. Kamel, M. Kindelan, and P. Sguazzero, p. 510.* Computerized seismic prospecting is an echo-ranging technique usually targeted at accurate mapping

of oil and gas reservoirs. In seismic surveys an impulsive source, often an explosive charge, located at the earth's surface generates elastic waves which propagate in the subsurface; these waves are scattered by the earth's geological discontinuities back to the surface, where an array of receivers registers the reflected signals. The data recorded are then processed in a complex sequence of steps. Among them, seismic migration and stacking velocity estimation represent two characteristic components of the process solving the inverse problem of recovering the structure and the physical parameters of the earth's geologic layers from echo measurements. A complementary tool in relating seismic data to the earth's inhomogeneities is provided by seismic numerical models, which assume a subsurface structure and compute the seismic data which would be collected in a field survey, by solving the direct problem of exploration geophysics. This paper describes a vectorized and parallelized implementation of a two-dimensional seismic elastic model on the IBM 3090 VF Vector Multiprocessor. An implementation of a parallel seismic migration algorithm is then described. The paper also reports performance data for a vector/parallel implementation on the IBM 3090 of some typical seismic velocity estimation algorithms. The three problems chosen are representative of a wide class of geophysical computations, and the results summarized in this paper show their suitability for efficient implementation on the IBM 3090 Vector Multiprocessor; combined vector/parallel speedups in the range 15–30 are in fact observed.

Effective utilization of IBM 3090 large virtual storage in the numerically intensive computations of *ab initio* molecular orbitals *by M. Sakaki, H. Samukawa, and N. Honjou, p. 528.* A new level of storage hierarchy, called Expanded Storage and available on the IBM 3090 system, is utilized by the MVS/XA™ operating system as high-speed paging equipment, allowing a user to hold application data in large virtual storage. To exploit the large virtual storage capability of the IBM 3090, a new application technique was developed for numerically intensive computations of *ab initio* molecular orbitals where high-speed transfer of a vast amount of intermediate data is a common requirement of most application programs. An application program running under MVS/XA was modified so that it could handle a vast amount of intermediate data in large virtual storage combined with Expanded Storage, achieving a 4- to 10-fold improvement in turnaround time at a CPU rate-determining step (SCF step) in medium-sized molecules.

PAM-CRASH on the IBM 3090/VF: An integrated environment for crash analysis *by P. Angeleri, D. F. Lozupone, F. Piccolo, and J. Clinckemaillie, p. 541.* PAM-CRASH® is an industrial code developed by Engineering Systems International (ESI) S.A. and designed specifically for automotive crashworthiness analysis. We discuss the problems encountered and describe the solutions provided for an efficient migration of the code on the IBM 3090/VF system. Runs on actual test cases have shown a vector/scalar speedup between 2.7 and 3.5. Moreover, we present the program modifications we have introduced in order to exploit parallel processing using the Multitasking Facility of the VS FORTRAN compiler. Performance results for 3090/VF systems, from the Model 200E to the Model 600E, are shown. Finally, we describe the restructuring of the graphic processors, PRE-3D and DAISY, to allow an effective use of the IBM 5080 Graphics System capabilities in providing an integrated design environment for crash analysis.

Interactive computations and display of characteristics of the radiation scattered by a sphere: A demonstration for PS/2 Model 80 *by P. Halpern and J. V. Dave, p. 561.* The Personal System/2® (PS/2®) Model 80 with its math coprocessor provides a considerable amount of computing power which can be used with advantage to solve technical problems interactively at a standalone workstation. To demonstrate this capability, a scientific program routinely used in diverse disciplines requiring significant computing power was modified to run on the PS/2 Model 80. It computes and displays variations of the specific intensity and degree of polarization of the electromagnetic radiation scattered by a sphere of given refractive index. For a sphere of size parameter of 100, about 475 thousand double-precision floating-point calculations are performed and the results displayed in graphic format in less than ten seconds. The selected algorithm is routinely used in several different disciplines such as astronomy, atmospheric optics, chemical engineering, colloidal chemistry, and remote sensing. Because it requires the evaluation of spherical Bessel functions with complex arguments, and derivatives of the Legendre polynomials, it was selected as a representative problem of numerically intensive computing.

Complete issues of the *IBM Systems Journal* may be ordered from an IBM branch office at $6.00 each for three years after publication by specifying the order number that is given on the back cover of an issue or in the Abstracts section of this index. Reprints of individual papers are available for five years and may be ordered from an IBM branch office at $1.00 each by specifying the order number that appears on the last page of a paper or is given in the Author Index section of this index. Additional copies of this index may be ordered for $1.00 each, using the order number on the back cover. (IBM employees in the U.S.A. can order all of the above from Mechanicsburg.) Copies of earlier issues and individual papers are available from University Microfilms, 300 N. Zeeb Road, Ann Arbor, Michigan 48106.

The *IBM Systems Journal* is abstracted by *Chemical Abstracts*, *Computer Abstracts*, *Computer & Control Abstracts*, *Computer & Information Systems*, *Computer Literature Index*, *Data Processing Digest*, *Ekspress Informatsiia*, *Electrical & Electronics Abstracts*, *The Engineering Index*, *Information Science Abstracts*, *Mathematical Reviews*, *New Literature on Automation (Netherlands)*, *Operations Research/Management Science*, *Referativnyi Zhurnal*, and *Science Citation Index*. Reviews appear in *Computing Reviews*.

G321-5344-00